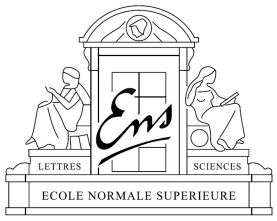
Primitives et constructions en cryptographie asymétrique

Damien Vergnaud



École normale supérieure
DI/ENS – École Doctorale de Sciences Mathématiques de Paris Centre

Primitives et constructions en cryptographie asymétrique

THÈSE D'HABILITATION

présentée pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches de l'École normale supérieure

(Spécialité Informatique)

par

Damien Vergnaud

Michel Abdalla ..... *Examinateur*
Jean-Claude Bajard ..... *Examinateur*
Gilles Barthe ..... *Rapporteur*
Dario Catalano ..... *Examinateur*
Jean-Sébastien Coron ..... *Examinateur*
Antoine Joux ..... *Rapporteur*
Fabien Laguillaumie ..... *Examinateur*
Kenny Paterson ..... *Rapporteur*
David Pointcheval ..... *Examinateur*
Jacques Stern ..... *Examinateur*

Travaux effectués au sein de l'Équipe de Cryptographie
du Département d'Informatique de l'École normale supérieure

À Louis

Cette habilitation est l'occasion pour moi de remercier chaleureusement toutes les personnes qui m'ont permis de progresser dans la conduite de mes travaux de recherche depuis 2001.

J'adresse mes remerciements les plus chaleureux aux membres du jury et en particulier aux rapporteurs qui m'ont fait l'honneur d'accepter mon invitation malgré des emplois du temps surchargés. Je suis très honoré que mon travail scientifique ait été évalué par de tels experts. Je remercie les différents chercheurs avec qui j'ai collaboré pour l'ensemble des échanges scientifiques que nous avons eus au cours de ces dernières années. Je tiens à remercier spécialement tous les étudiants avec qui j'ai eu le plaisir de travailler et en particulier mes doctorants qui ont accepté de tenter l'aventure avec moi. Les travaux présentés dans ce manuscrit doivent beaucoup à leurs idées. Je les remercie tous aussi pour les moments de convivialité que nous avons partagés en marge de nos travaux scientifiques. Je tiens également à exprimer ma gratitude à tous les techniciens, secrétaires et ingénieurs que j'ai pu côtoyer au sein des différents laboratoires et universités que j'ai fréquentés. Je tiens enfin à remercier mes collègues du département d'informatique de l'ENS (et d'ailleurs) pour le travail pédagogique et administratif que nous menons ensemble.

Selon la formule en usage, ces personnes sont trop nombreuses pour être citées mais ce n'est pas une raison pour ne pas le faire ... Je souhaite donc exprimer toute ma reconnaissance à Michel Abdalla, Laila El Aimani, Ali Akhavi, Roberto Amadio, Magdi Amer, Francesco Amoroso, Michelle Angely, Fayçal Aouad, Nuttapong Attrapadung, Abdelhak Azhari, Abdelmalek Azizi, Jean-Claude Bajard, Gilles Barthe, Aurélie Bauer, Jacques Beigbeder, Sonia Belaid, Mostafa Belkasmi, Mihir Bellare, Hussain Benazza, Fabrice Benhamouda, Valérie Berthé, Julien Bertrane, Raghav Bhaskar, Lise-Marie Bivard, Jean-Luc Blanc, Bruno Blanchet, Olivier Blazy, Charles Bouillaguet, Anne Bouillard, John Boxall, Xavier Boyen, Emmanuel Bresson, Romain Brette, David Cadé, Sébastien Canard, Christophe de Cannière, Angelo De Caro, Guilhem Castagnos, Dario Catalano, Julien Cathalo, Pierre-Louis Cayrel, Yuanmi Chen, Céline Chevalier, Benoît Chevallier-Mames, Marc De Crisenoy, Gérard Cohen, Iwen Coisel, Hubert Comon-Lundh, Éric Colin de Verdière, Mario Cornejo, Jean-Sébastien Coron, Patrick Cousot, Guillaume Dabosville, Léonard Dallot, Isabelle Delais, Cécile Delerablée, Patrick Derbez, Jérémie Detrey, Julien Devigne, Itai Dinur, Yevgeniy Dodis, Éric Domenjoud, Renaud Dubois, Vivien Dubois, Léo Ducas, Orr Dunkelman, Sylvain Duquesne, Maribel Fernandez, Dario Fiore, Emmanuel Fouotsa, Pierre-Alain Fouque, Georg Fuchsbauer, David Galindo, Nicolas Gallot, Nicolas Gama, Sanjam Garg, Pierrick Gaudry, Joachim von zur Gathen, Valérie Girardin, Marc Girault, Louis Goubin, Aline Gouget, Eleonora Guerrini, Cheikh Gueye, Cédric Guillard, Aurore Guillevic, Nicolas Gürel, Brett Hemenway, Javier Herranz, Emeline Hufschmitt, Laurent Imbert, Sylvia Imbert, Sorina Ionica, Joëlle Isnard, Malika Izabachène, Amandine Jambert, Damien Jamet, Mohamed Jaoua, Jérémy Jean, Antoine Joux, Marc Joye, Eike Kiltz, Salah Labhalla, Patrick Lacharme, Pauline Lafitte, Pascal Lafourcade, Fabien Laguillaumie, Hélène Lanéry, Adeline Langlois, Yann Lefeuvre, Isabelle Lenormand, Tancrède Lepoint, Reynald Lercier, Gaëtan Leurent, Roch Lescuyer, Eric Levieil, Zhentao Li, Benoît Libert, Pierre Loidreau, Satya Lokam, Vadim Lyubashevsky, Stéphane Mallat, Louis Mandel, Mark Manulis, Joana Marim, Claire Mathieu, Thierry Mefenza, Nicolas Méloni, Antoine Miné, Valérie Mongiat, Jean Monnerat, Annick Montanvert, Paz Morillo, Nadia El Mrabet, David Naccache, Phong Nguyen, Abderrahmane Nitaj, Rafail Ostrovsky, Ayoub Otmani, Pascal Paillier, Miriam Paiola, Kenny Paterson, Alain Passelègue, Thomas Peters, Thomas Peyrin, Alex Pfister, Duong Hieu Phan, Thomas Plantard, Michel Pocchiola, David Pointcheval, Jean Ponce, Marc Pouzet, Thomas Prest, Emmanuel Prouff, Elizabeth Quaglia, Jean-Jacques Quisquater, Michael Quisquater, Carla Ràfols, Christian Rechberger, Oded

Regev, Philippe Renevier-Gonin, Jean-Sébastien Revy, Éric Reyssat, Ludovic Ricardou, Vincent Rijmen, Sylvain Ruhault, Oliviers Sanders, Nicolas Sendrier, Yannick Seurin, Ibra Seye, Adi Shamir, Jamshid Shokrollahi, Hervé Sibert, Benjamin Smith, Djiby Sow, Jacques Stern, Mario Strefler, Sabrina Tarento, Adrian Thillard, Mehdi Tibouchi, Jacques Traoré, Carole Trochu, Brigitte Vallée, Serge Vaudenay, Pascal Véron, Marion Videau, Alfredo Viola, Marie Virat, Jean Vuillemin, Laurent Vuillon, Daniel Wichs, Hoeteck Wee, David Xiao, Jean-Christophe Zapalowicz et Sébastien Zimmer.

J'ai enfin une pensée particulière pour ma famille et mes amis et mes remerciements les plus tendres seront pour Juliette, pour sa présence indispensable, jour après jour, et pour Louis, qui irradie un bonheur communicatif.

Contents

Part I

Primitives and Constructions in Asymmetric Cryptography

Chapter 1

Introduction

This thesis presents the research work done by the author (and several co-authors) since his doctorate thesis. To respect space constraints and retain the focus on our main research theme, only works related to the design and analysis of primitives and protocols in public-key cryptography are presented. Our research works in other domains of cryptography (*e.g.* pseudo-random generator analysis [DPR+13,FVZ13,BVZ12], elliptic and hyperelliptic cryptography [JTV10,GV12] or multi-party computation [Ver11]) and outside cryptography are not presented. A complete list of personal publication is presented on page 59.

1.1 Reductionist Security

The basic task in cryptography is to enable to parties to communicate "securely" over an insecure channel, in a way that guarantees (for instance) confidentiality, integrity and authenticity of their communication (among other possible security goals). The design of cryptographic protocols in order to achieve these goals is a delicate, error-prone and difficult task. Indeed, since the introduction of public-key cryptography, many cryptographic schemes have been designed and a significant proportion have thereafter been broken. In particular, the fact that a cryptographic algorithm withstood cryptanalytic attacks for several years should not be considered as a kind of validation procedure.

The idea of provable security was introduced thirty years ago in the pioneering work of Goldwasser and Micali [GM84] (for which they received the Turing award in 2013). Their approach relies on the principle that the security of cryptographic schemes is proven secure based on mathematically precise assumptions. These assumptions can be general (such as the existence of one-way functions or trapdoor one-way functions) or specific (such as the hardness of the discrete logarithm problem in specific group families). The security argument is a *reduction* (in the complexity theory meaning) that transforms any *adversary* $\mathcal{A}$ against a cryptographic protocol into an algorithm (formally a probabilistic (polynomial-time) Turing machine) that breaks the underlying assumption (*i.e.* that solves the underlying mathematical problem).

The first step in this approach is to define formally what is an adversary against a cryptographic protocol. This definition is divided into two parts; a security model that specifies what it means for a protocol to be "secure", and an adversarial model that specifies what powers an adversary attacking the protocol is allowed to possess An adversary is then modelled as a probabilistic Turing machine attempting to fulfil the goal while given access to these resources when interacting with the cryptographic scheme. A reductionist security proof for some cryptographic protocol Π to some (alleged) hard mathematical problem $\mathcal{P}$ is then an algorithm $\mathcal{R}$, called the reduction, for solving $\mathcal{P}$ given access to a hypothetical algorithm $\mathcal{A}$ that breaks this security definition. In other words, the reduction shows that the only way to defeat the protocol is to

(implicitly) break the underlying computational problem.

To quantify this statement, a variable k termed the *security parameter* is usually used to measure the input sizes of the mathematical problem $\mathcal{P}$ and the cryptographic protocol Π. The resource requirements of Π as well as the adversary $\mathcal{A}$ probability of breaking security are expressed in terms of k. In *asymptotic security*, a scheme is deemed secure if for all probabilistic polynomial-time (in k) adversaries $\mathcal{A}$ with a noticeable probability of success $\varepsilon(k)$ in breaking Π (i.e. $\varepsilon(k) = \Omega(k^{-n})$ for some integer $n \in \mathbb{N}$), the reduction $\mathcal{R}$ is a probabilistic polynomial-time (in k) algorithm with a noticeable probability of success in solving $\mathcal{P}$. *Concrete security* is a practice-oriented approach that aims to give precise estimates of the computational complexity and success probability of $\mathcal{R}$ in function of those of $\mathcal{A}$.

This paradigm has been extremely successful and many cryptographic tasks have been put under rigorous treatment and realized under a number of well-studied complexity-theoretic intractability assumptions. It is worth noting that security proofs give no assurance of security against adversaries that are not described by the security model used (*e.g.* measurement of side-channel information) or if the underlying mathematical assumption turns out to be wrong.

All cryptographic protocols presented in this document have been analyzed in the framework of "reductionist security" (with a concrete security approach). The rest of this expository chapter is devoted to the presentation of the tools necessary for their analysis. We tried to minimize the use of the *random oracle model* formalized by Bellare and Rogaway in 1993 [BR93]. In this idealized model, cryptographic protocols are designed and proved secure under the additional assumption that publicly available functions that are chosen truly at random exist. In the security reduction, these random oracles can only be accessed by the adversary in a black-box way, by providing an input and obtaining the corresponding output. The random oracle model has been used to prove the security of numerous cryptosystems, and it has lead to simple and efficient designs that are widely used in practice. This mathematical abstraction is useful but no concrete function can implement a true random oracle. Numerous papers have shown artificial schemes that are provably secure in the random oracle model, but completely insecure when any real function is substituted for the random oracle [CGH04].

We present some computational assumptions (related to the discrete logarithm problem) in Section 1.2). We then present security models (and efficient protocols) for two basic primitives of asymmetric cryptography – public-key encryption schemes and digital signature schemes – in Section 1.3 and 1.4 respectively. Finally, we briefly describe the *universal composability* framework (Section 1.5) that allows for modular design and analysis of complex cryptographic protocols from relatively simple building blocks.

1.2 Structures and Computational Assumptions

In this thesis, all cryptographic constructions that we present are defined either in (multiplicative) groups $\mathbb{G}$ (of prime order p and generator g, that we will denote $(p, \mathbb{G}, g)$) or in *bilinear structures*, which are defined as follows:

Definition 1.2.1 [Bilinear Groups] A bilinear structure is a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, generated respectively by g_1,g_2 and $e(g_1, g_2)$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear form, i.e. :

$$\forall X \in \mathbb{G}_1, \forall Y \in \mathbb{G}_2, \forall \lambda, \mu \in \mathbb{Z}_p : e(X^\lambda, Y^\mu) = e(X, Y)^{\lambda\mu}$$

and $e(g_1, g_2)$ does indeed generate the prime order group $\mathbb{G}_T$.

Such groups are commonly instantiated on elliptic curves on which such pairings can be defined as bilinear forms. Galbraith *et al.* [GPS08] have split such instantiations in three main types:

- Type-I, where $\mathbb{G}_1 = \mathbb{G}_2$, and $g_1 = g_2$, those groups are said to be symmetric and can be simplified as $(p, \mathbb{G}, \mathbb{G}_T, e, g)$,
- Type-II, if there exists a computationally efficient homomorphism from $\mathbb{G}_2$ in $\mathbb{G}_1$, but none from $\mathbb{G}_1$ to $\mathbb{G}_2$,
- Type-III, if such efficient homomorphism does not exist in either direction.

The Type-I instantiation was popular among cryptographers for a long time since it simplifies the presentation of protocols. However, with the recent advances on discrete logarithm in multiplicative groups of finite fields due to Joux *et al.* (*e.g.* [Jou13, JP13, BGJT14]), this instantiation becomes very inefficient. Therefore, we will present protocols using the Type-III instantiation since they are the more general and the more efficient in practice.

These structures $(p, \mathbb{G}, g)$ or $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ are generated by appropriate polynomial-time algorithms that given a security parameter $k \in \mathbb{N}$ (usually viewed as a unary input 1^k), outputs a description of such a structure (of polynomial size in k) in which solving some computational or decisional assumptions are (strongly) supposed to require exponential time in k (e.g. $O(2^k)$). We now describe several computational problems on which the security of our constructions will rely (in a group $(p, \mathbb{G}, g)$):

Definition 1.2.2 [Discrete Logarithm (DL)] The Discrete Logarithm hypothesis says that given $(p, \mathbb{G}, g)$, and an element $h \in \mathbb{G}$, picked uniformly at random, it is hard to find $\mu \in \mathbb{Z}_p$ such that $h = g^\mu$.

Definition 1.2.3 [Decisional Linear (DLin [BBS04])] The Decisional Linear hypothesis says that in a multiplicative group $(p, \mathbb{G}, g)$ when we are given $(g^\lambda, g^\mu, g^{\alpha\lambda}, g^{\beta\mu}, g^\psi)$ for unknown random $\alpha, \beta, \lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \alpha + \beta$.

Definition 1.2.4 [Decisional Diffie Hellman (DDH [Bon98])] The Decisional Diffie-Hellman hypothesis states that in a multiplicative group $(p, \mathbb{G}, g)$, given (g^μ, g^ν, g^ψ) for unknown $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \mu\nu$.

We can consider variants of these problems in bilinear structures. One can see readily that the Decisional Diffie-Hellman problem is tractable in Type-I bilinear structures thanks to the bilinear map e. However in Type-II and Type-III bilinear structures, it makes sense to consider the following problems:

Definition 1.2.5 [External Diffie Hellman in $\mathbb{G}_1$ (XDH [BBS04])] This variant of the previous hypothesis states that in a Type-II bilinear group, given $(g_1^\mu, g_1^\nu, g_1^\psi)$ for unknown $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \mu\nu$. (In other words DDH is hard in $\mathbb{G}_1$.) A variant can say that DDH is hard in $\mathbb{G}_2$.

Definition 1.2.6 [Symmetric External Diffie Hellman (SXDH [ACHdM05])] This last variant, used in Type III bilinear groups, states that DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

We also describe two computational hypotheses related to the DDH:

Definition 1.2.7 [Computational Diffie Hellman (CDH [DH76])] The Computational Diffie-Hellman hypothesis states that in a multiplicative group $(p, \mathbb{G}, g)$, given (g^μ, g^ν) for unknown $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$, it is hard compute $g^{\mu\nu}$.

Definition 1.2.8 [Extended Computational Diffie-Hellman problem (CDH^+ [BFPV11]):] Let us be given two (multiplicative) groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order p with (g_1, g_2) as respective generators. The CDH^+ assumption states that given $(g_1, g_2, g_1^\mu, g_2^\mu, g_1^\nu)$, for random $\mu, \nu \in \mathbb{Z}_p$, it is hard to compute $g_1^{\mu\nu}$.

1.3 Encryption schemes

The classical goal of a public-key encryption scheme is to preserve the privacy of messages: an adversary should not be able to learn from a ciphertext information about its plaintext beyond the length of that plaintext. In this section, we provide a formal security definition (a notion called *semantic security* or *indistinguishability of ciphertexts* [GM84]) capturing this intuitive statement.

1.3.1 Definition

An encryption scheme is defined by four algorithms

$$(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}) :$$

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public (encryption) key ek and the private (decryption) key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ outputs the plaintext m encrypted in c.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pairs $(\mathsf{ek}, \mathsf{dk})$ output by $\mathsf{KeyGen}_\mathcal{E}(\mathsf{param})$ and all messages m we have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m)) = m$.
- *Indistinguishability under chosen-plaintext attacks* ($\mathsf{IND}-\mathsf{CPA}$): this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ is permitted to keep some internal state between the various calls `FIND` and `GUESS`.

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek})$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*)$
6. `RETURN` b'

The advantages are

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(k) &= \Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(k) = 1] \\ \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k,t) &= \max_{\mathcal{A} \le t} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(k). \end{aligned}$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(k)$ for $b \in \{0,1\}$ runs in time at most t. The scheme $\mathcal{A}$ is deemed $\mathsf{IND}-\mathsf{CPA}$-secure, if for all polynomials p, $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k, p(k))$ is a negligible function of k (i.e. asymptotically smaller than the inverse of any polynomial in k).

One might want to increase the requirements on the security of an encryption, in this case the $\mathsf{IND}-\mathsf{CPA}$ notion can be strengthened into Indistinguishability under Adaptive Chosen Ciphertext Attack $\mathsf{IND}-\mathsf{CCA2}$. The non-adaptive notion was introduced in [NY90], while the adaptive one was introduced a year later in [RS91]:

- *Indistinguishability under chosen-ciphertext attacks* ($\mathsf{IND}-\mathsf{CCA2}$): This notion states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and can ask several decryption of ciphertexts as long as they are not the challenge one.

$\mathsf{Exp}^{\mathsf{ind}-\mathsf{cca}-b}_{\mathcal{E},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. `IF` $(c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

  - Where the $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ (initially empty) of decrypted ciphertexts.

In some contexts (*e.g.* password-based authenticated key exchange – see Section 4.4), it is useful to consider the notion of *labelled encryption*, where the message M is encrypted but with some extra public information ℓ. This label can be useful to include session information for example.

A labelled public-key encryption scheme is defined by four algorithms:

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key pk and the decryption key dk;
- $\mathsf{Encrypt}(\ell, \mathsf{pk}, m; \rho)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key pk, using the random coins ρ;
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext M encrypted in c under the label ℓ, or $\perp$.

The correctness and the indistinguishability security notions for labelled Encryption Scheme are defined in a similar way (but with some subtleties, see Section H.A.1) .

1.3.2 ElGamal encryption [Gam85]

ElGamal encryption in a (cryptographic) group $(p, \mathbb{G}, g)$ is defined by the four algorithms.

- $\mathsf{Setup}(1^k)$: The scheme needs a group $(p, \mathbb{G}, g)$, where $\mathbb{G}$ is a group of prime order p with generated by g.
- $\mathsf{KeyGen}(\mathsf{param})$: One chooses a scalar α uniformly at random in $\mathbb{Z}_p$ which defines $U = g^\alpha$. The public (encryption) key is $\mathsf{ek} = U$ and the private (decryption) key is $\mathsf{dk} = \alpha$;

- $\mathsf{Encrypt}(\mathsf{ek} = U, m; r)$: The algorithm is given a message $m \in \mathbb{G}$, the public (encryption) key is $\mathsf{ek} = U$ and a random $r \in \mathbb{Z}_p$ and publishes $c = (c_1 = m \cdot U^r, c_2 = g^r)$.
- $\mathsf{Decrypt}(\mathsf{dk} = \alpha, c = (c_1, c_2))$: The algorithm computes $m = c_1/c_2^{\alpha}$.

This scheme is semantically secure against chosen-plaintext attacks ($\mathsf{IND} - \mathsf{CPA}$) assuming the hardness of DDH in the underlying group families $\{(p, \mathbb{G}, g)\}$. Moreover, it has some nice homomorphic properties: it allows multiplication to be carried out on ciphertexts in such a way that it generates an encrypted result which, when decrypted, gives the product performed on the corresponding plaintext. With this scheme, it is possible to re-randomize a ciphertext c to a new ciphertext c' such that c and c' encrypt the same plaintext but are statistically independent. These properties will find useful applications in Chapter 2.

1.3.3 Commitment

Commitments allow a user to commit to a value without revealing it, but without the possibility to later change his mind. A commitment scheme is composed of three algorithms:

- $\mathsf{Setup}(1^k)$ generates the system parameters, according to the security parameter k;
- $\mathsf{Commit}(m; r)$ produces a commitment c on the input message $m \in \mathcal{M}$ using the random coins $r \xleftarrow{\$} \mathcal{R}$;
- $\mathsf{Decommit}(c, m; w)$ opens the commitment c and reveals the message m, together with a witness w that proves the correct opening.

Such a commitment scheme should be both *hiding*, which says that the commit phase does not leak any information about m, and *binding*, which says that the decommit phase should not be able to open to two different messages. Additional features are also sometimes required, such as non-malleability, extractability, and/or equivocability. As for labelled encryption, we may also include a label ℓ, which is an additional public information that has to be the same in both the commit and the decommit phases (see Section D.2 for formal definitions).

1.4 Signature schemes

Digital signatures are one of the most useful and fundamental primitives resulting from the invention of public-key cryptography. They are the electronic version of handwritten signatures for digital documents: a user's signature on a message m is a string which depends on m, on the signer's public (and secret) key and, possibly, on random coins. The validity of the signature can be checked by using the signer's public key only. The intuitive security notion would be the impossibility to forge signatures without the knowledge of the secret key even after seeing signatures on messages of his choice (it has been formalized in [GMR88]).

1.4.1 Digital Signatures

Signature scheme. A signature scheme is defined by four algorithms

$$(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif}) :$$

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;

- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public (verification) key vk and the private (signing) key sk;
- $\mathsf{Sign}(\mathsf{sk}, m; s)$ produces a signature σ on the input message m, under the signing key sk, and using the random coins s;
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

A signature scheme $\mathcal{S}$ should satisfy the following properties

- *Correctness*: for all key pairs $(\mathsf{vk}, \mathsf{sk})$ and all messages m we have $\mathsf{Verif}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$.
- *Existential unforgeability under (adaptive) chosen-message attacks [GMR88]*: this security notion can be formalized by the following security game, where it makes use of the oracle $\mathtt{Sign}$:
  - $\mathtt{Sign}(\mathsf{sk}, m)$: This oracle outputs a valid signature on m under the signing key sk. The input queries m are added to the list $\mathcal{SM}$.

  The success probabilities are

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{S,\mathcal{A}}(k) = 1] \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$$

  where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$ runs in time at most t. The scheme $\mathcal{S}$ is deemed $\mathsf{EUF-CMA}$-secure, if for all polynomial p, $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, p(k))$ is a negligible function of k (i.e. asymptotically smaller than the inverse of any polynomial in k).

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathtt{Sign}(\mathsf{sk}, \cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. `IF` $M \in \mathcal{SM}$ `RETURN` 0
6. `ELSE RETURN` b

1.4.2 Waters signatures

Waters signatures [Wat05] form a simple and efficient digital signature scheme in bilinear structures. They were proposed in the context of Type-I structures and the existential unforgeability of the scheme can be proved under the computational Diffie-Hellman (CDH) assumption. For efficiency reasons, we consider an asymmetric variant of Waters signatures we introduced in [BFPV11]:

Waters signature (in an asymmetric structure).

- $\mathsf{Setup}(1^k)$: in a bilinear structure $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, one chooses a random vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We also need an extra generator $h_1 \xleftarrow{\$} \mathbb{G}_1$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, \vec{u})$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = g_2^x$, and the secret key as $\mathsf{sk} = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g_1^s, \sigma_3 = g_2^s)$.

- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g_2) = e(h_1, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_3)$, and $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the CDH^+ assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$.

1.4.3 Blind Signatures

The issue of anonymity in electronic transactions was introduced for e-cash and e-mail in the early 1980's by Chaum, with the famous primitive of blind signatures [Cha82, Cha83]. These define an interactive signature protocol between a user and a signer, guaranteeing that the signed message, and even the resulting signature, are unknown to the signer; this property is called *blindness*. More precisely, if the signer runs several executions of the protocol leading to several message/signature pairs, he cannot link a pair to a specific execution: the view of the signer is unlinkable to the resulting message/signature pair. This unlinkability can be either computational, in which case we talk about *computational blindness*, or information-theoretic, we then talk about *perfect blindness*. The second security property for blind signatures is a notion of unforgeability, which has been formalized by Pointcheval and Stern [PS00] motivated by the use of blind signatures for e-cash: a user should not be able to produce more message/signature pairs (coins) than the number of signing executions with the bank (withdrawals). More recently, Schröder and Unruh [SU12] revisited the security model for other contexts.

Definition 1.4.1 [Blind Signature Scheme]
A blind signature scheme is defined by three algorithms $(\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSVerif})$ and one interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{U}\rangle$:

- $\mathsf{BSSetup}(1^k)$, where k is the security parameter, generates the global parameters param of the system;
- $\mathsf{BSKeyGen}(\mathsf{param})$ generates a pair of keys, the public (verification) key vk and the private (signing) key sk;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$: this is an interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, m)$, for a message $m \in \{0,1\}^n$. It generates a signature σ on m under vk related to sk for the user.
- $\mathsf{BSVerif}(\mathsf{vk}, m, \sigma)$ outputs 1 if the signature σ is valid with respect to m and vk, 0 otherwise.

As mentioned above, a blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: blindness and unforgeability.

Blindness states that a malicious signer should be unable to decide which of two messages m_0, m_1 has been signed first in two *valid* executions with an honest user.
Note that the malicious signer $\mathcal{A}$ can choose arbitrarily the keys and thus the verification key vk given to users. However, if $\mathcal{A}$ refuses to sign one of the inputs (i.e. $\sigma_i = \perp$ for $i \in \{0,1\}$) or if one of the signatures is invalid (i.e. $\mathsf{BSVerif}(\mathsf{vk}, m_i, \sigma_i) = 0$ for $i \in \{0,1\}$) then the two resulting signatures are set to $\perp$; the adversary therefore does not gain any advantage if he decides to prevent the normal game execution.

$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk}, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $\sigma_b \leftarrow \mathsf{BSProtocol}\langle\mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$
4. $\sigma_{1-b} \leftarrow \mathsf{BSProtocol}\langle\mathcal{A}, \mathcal{U}(\mathsf{vk}, m_{1-b})\rangle$
5. $b^* \leftarrow \mathcal{S}^*(\texttt{GUESS} : \sigma_0, \sigma_1)$;
6. $\texttt{RETURN}\ b^* = b$.

The advantages are

$$\begin{aligned}\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) &= \Pr[\mathsf{Exp}^{\mathsf{bl}-1}_{\mathcal{BS},\mathcal{A}}(k)=1] - \Pr[\mathsf{Exp}^{\mathsf{bl}-0}_{\mathcal{BS},\mathcal{A}}(k)=1]\\ \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS}}(k,t) &= \max_{\mathcal{A}\leq t} \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E},\mathcal{A}}(k).\end{aligned}$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$ for $b \in \{0,1\}$ runs in time at most t. The scheme $\mathcal{BS}$ is deemed blind, if for all polynomials p, $\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E}}(k,p(k))$ is a negligible function of k.

An adversary against the (one-more) unforgeability tries to generate $q+1$ valid signatures after at most q complete interactions with the honest signer. This security notion can be formalized by the security game $\mathsf{Exp}^{\mathsf{omuf}}_{\mathcal{BS},\mathcal{U}^*}(k)$ where the adversary is permitted to keep some internal state between the various calls $\texttt{INIT}_i$ (for $i \in \{1,\ldots,q_s\}$), `FIND` and `GUESS`.
The success probabilities are

$\mathsf{Exp}^{\mathsf{omuf}}_{\mathcal{BS},\mathcal{A}}(k)$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. For $i = 1,\ldots,q_s$, $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{A}(\texttt{INIT}_i : \mathsf{vk})\rangle$
4. $((m_1,\sigma_1),\ldots,(m_{q_s+1},\sigma_{q_s+1})) \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{vk})$;
5. `IF` $\exists i \neq j, m_i = m_j$ `OR` $\exists i, \mathsf{Verif}(\mathsf{pk}, m_i, \sigma_i) = 0$ `RETURN` 0
6. `ELSE RETURN` 1

$$\mathsf{Succ}^{\mathsf{omuf}}_{\mathcal{BS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{omuf}}_{\mathcal{BS},\mathcal{A}}(k)=1] \quad \mathsf{Succ}^{\mathsf{omuf}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A}\leq t} \mathsf{Succ}^{\mathsf{omuf}}_{\mathcal{S},\mathcal{A}}(k)$$

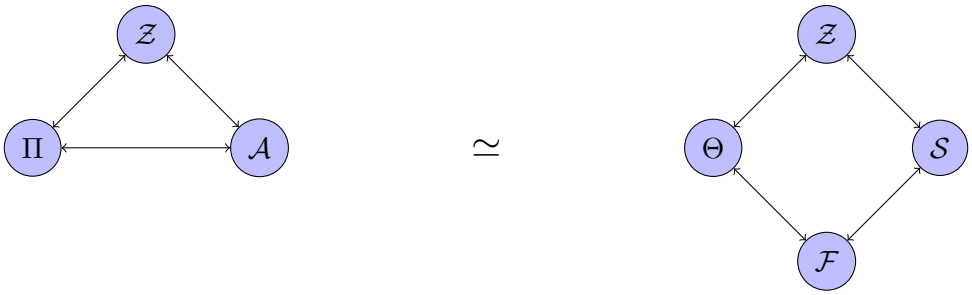
where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{omuf}}_{\mathcal{S},\mathcal{A}}(k)$ runs in time at most t. The scheme $\mathcal{S}$ is deemed $\mathsf{OMUF}-\mathsf{CMA}$-secure, if for all polynomial p, $\mathsf{Succ}^{\mathsf{omuf}}_{\mathcal{S}}(k,p(k))$ is a negligible function of k.

Concurrency in the context of blind signatures was put forth by Juels, Luby and Ostrovsky [JLO97] who presented the first security model for blind signatures that takes into account that the adversary may launch many concurrent sessions of the blind signing protocol (operating as either the user or the signer). In this document, we consider only *round-optimal* blind signatures (i.e. the user sends a single message to the signer and gets a single response) which are concurrently secure.

1.5 Universal Composability Framework

The Universal Composability (UC) framework introduced by Canetti [Can01] is a popular security paradigm. It guarantees that a protocol proven secure in this framework remains secure even if it is run concurrently with arbitrary —even insecure— protocols (whereas classical definitions only guarantee its security in the stand-alone setting). The UC framework enables one to split the design of a complex protocol into that of simpler sub-protocols.

In the context of multi-party computation, one wants several users P_i with inputs x_i to be able to compute a specific function $f(x_1,\ldots,x_n) = (y_1,\ldots,y_n)$ without leaking anything except y_i to P_i. Instead of following the classical approach which aims at listing exhaustively all the expected properties, Canetti did something else and tried to define how a protocol should ideally



Real world Ideal world

Protocol Π is a UC-secure realization of functionality $\mathcal{F}$ if the real interaction (left) is indistinguishable from the ideal interaction (right).
Θ indicates the party running the ideal protocol that simply relays messages between $\mathcal{F}$ and $\mathcal{Z}$.

Figure 1.1: Universal Composability

work: what are the inputs, and what are the available outputs. For that, he specified two worlds: the real world, where the protocol is run with some possible attacks, and the ideal world where everything would go smoothly, and namely no damage can be done with the protocol. For a good protocol instantiation, it should be impossible to distinguish, for an external player, the real world from the ideal one.

In the *ideal world* there is indeed an incorruptible entity named the *ideal functionality*, to which players can send their inputs privately, and then receive the corresponding outputs without any kind of communication between the players. This way the functionality can be set to be correct, without revealing anything except what is expected. It is thus perfectly secure. A protocol, in the *real world* with real players and thus possibly malicious players, should create executions that look similar to the ones in the previous world. This is to show that the communication between the players should not give more information than the functionality's description and its outputs.

As a consequence, the formal security proof is performed by showing that for any external entity, that gives inputs to the honest players and gets the outputs but that also controls the adversary, the executions in the two above worlds are indistinguishable. More concretely, in order to prove that a protocol $\mathcal{P}$ realizes an ideal functionality $\mathcal{F}$, we consider an environment $\mathcal{Z}$ which can choose inputs given to all the honest players and receives back the outputs they get, but which also controls an adversary $\mathcal{A}$. Its goal is to distinguish in which case it is: either the real world with concrete interactions between the players and the adversary, or the ideal world in which players simply forward everything to and from the ideal functionality and the adversary interacts with a simulator $\mathcal{S}$ to attack the ideal functionality. We have to build a simulator $\mathcal{S}$ that makes the two views indistinguishable to the environment: since the combination of the adversary and the simulator cannot cause any damage against the ideal functionality, this shows that the adversary cannot cause any damage either against the real protocol.

The main constraint is that the simulator cannot rewind the execution as often done in classical proofs, since it interacts with an adversary under the control of the environment: there is no possible rewind in the real word, it is thus impossible too in the ideal world.

The adversary $\mathcal{A}$ has access to the communication but nothing else, and namely not to the inputs/ouputs for the honest players. In case of corruption, it gets complete access to inputs and the internal memory of the honest player, and then gets control of it.

The composition theorem [Can01] forms the crux of the UC-security framework. It establishes the cryptographic equivalence of protocols which emulate one another. The construction of arbitrarily complex UC-secure protocols from basic protocols for cryptographic primitives follows directly from the validity of the composition operation. The composition theorem notably describes how a theoretical interface with an ideal functionality (via the ideal protocol) can be replaced with a secure real-world protocol.

Chapter 2

Malleable Cryptography

As mentioned in the introduction, the basic goal of an encryption scheme is to guarantee the privacy of data and a good security definition is the notion of semantic security as defined by Goldwasser and Micali [GM84]. When encryption schemes are deployed in more complex environments, the demands for security of encryption grow beyond just the basic privacy requirement. In [DDN91, DDN00], Dolev, Dwork and Naor defined the notion of *non-malleability*. This ensures that it is infeasible for an adversary to modify a vector of ciphertexts $(c_1, \ldots, c_n)$ into other ciphertexts of messages which are related to the decryption of $c_1, \ldots, c_n$. This stronger notion of security is critical for many practical applications. This security notion was studied in numerous papers (*e.g.* [Sah99, BS99]) and strengthened by Fischlin [Fis05].

The notion of non-malleability was then applied successfully to various cryptographic primitives such as commitments (*e.g.* [DIO98, FF00, DKOS01]), zero-knowledge proofs [DDN91, DDN00, PR05] or multi-party computation (to prevent man-in-the-middle attacks).

On the other hand, it has been realized that, in specific settings, malleability in cryptographic protocols can actually be a very useful feature. The notion of *homomorphic encryption* allows specific types of computations to be carried out on ciphertext and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. Among the classical homomorphic encryption schemes, one can find the Goldwasser-Micali encryption scheme [GM84], ElGamal encryption scheme [Gam85] or Paillier's encryption scheme [Pai99]. Until recently, all the homomorphic encryption schemes were able to perform only one operation (addition or multiplication) on ciphertexts (a notable exception being the scheme by Boneh, Goh and Nissim [BGN05]). In 2009, Gentry proposed the first *fully* homomorphic encryption scheme in 2009 [Gen09]. His scheme (and subsequent improvements) supports both addition and multiplication and therefore any circuit can be homomorphically evaluated on ciphertexts. The homomorphic property can be used to create secure voting systems, collision-resistant hash functions, private information retrieval schemes, and – for fully homomorphic encryption – enables widespread use of cloud computing by ensuring the confidentiality of processed data.

Recently, it has been shown that malleability is an interesting feature for other primitives (such as, counter-intuitively, signatures or proof systems). In this chapter, we briefly present several applications of malleability for encryption schemes and signature schemes [HLOV11, IPV10, LV11, BFPV11, BPV12a, BFPV13, LV08c, LV08b, LV08a]. We also present constructions achieving strong security guarantees and primitives that found applications when implemented with a suitable malleable proof system that we will describe in the next chapter.

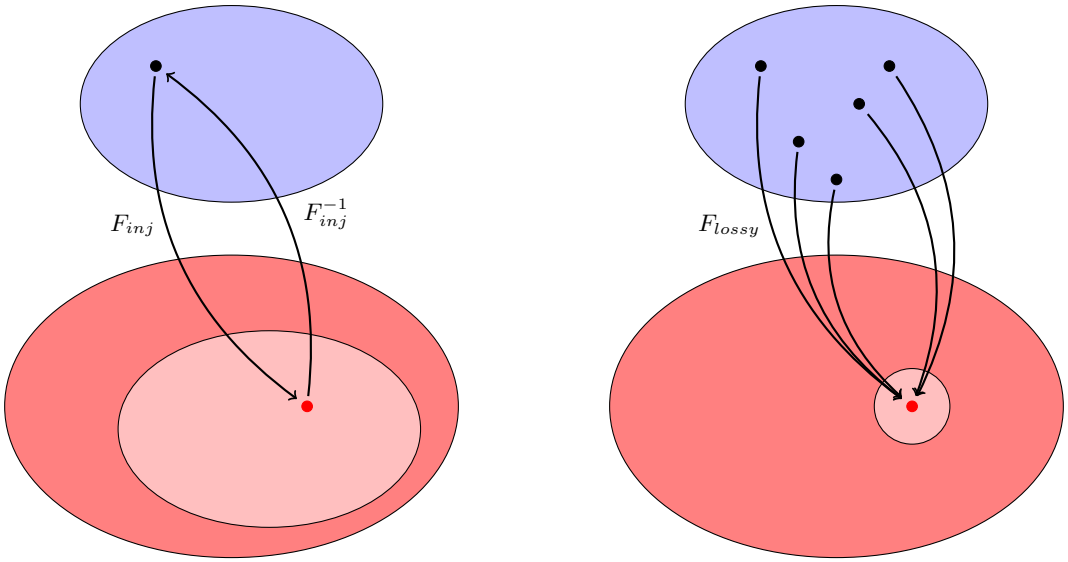


Figure 2.1: Lossy Trapdoor Functions

2.1 Malleability in Encryption Schemes

2.1.1 Lossy Encryption

In [GOS06a] Groth, Ostrovsky and Sahai introduced the notion of "parameter switching" technique in encryption keys. In particular, they defined homomorphic commitments that allow parameter switching in the key generation to allow either producing perfectly hiding or perfectly binding keys, with the requirement that it is computationally indistinguishable to tell which of the two modes is being used. This "parameter switching" technique proved incredibly useful in cryptography. The technique was also named (and renamed) several times.

In [PW08], Peikert and introduced a new primitive called *lossy trapdoor functions*, which is a family of functions F that are created to behave in one of two modes. The first mode samples a function F_{inj} that matches the usual completeness condition for an (injective) trapdoor function: given a suitable trapdoor for F_{inj} (denoted F_{inj}^{-1}, the entire input x can be efficiently recovered from $F_{inj}(x)$. In the second mode, the sampled functions F_{lossy} statistically lose a significant amount of information about its input, *i.e.*, every output of F_{lossy} has many preimages (see Figure 2.1). As in "parameter switching", it is computationally indistinguishable to tell which of the two modes is being used.

In [PVW08], Peikert, Vaikuntanathan and Waters defined Dual-Mode Encryption, a type of cryptosystem with two types public-keys, injective keys on which the cryptosystem behaves normally and "lossy" or "messy" keys on which the system loses information about the plaintext. In particular they require that the encryptions of any two plaintexts under a lossy key yield distributions that are statistically close, yet injective and lossy keys remain computationally indistinguishable. In [BHY09] Bellare, Hofheinz and Yilek define *Lossy Encryption*, expanding on the definitions of Dual-Mode Encryption in [PVW08], and Meaningful/Meaningless Encryption in [KN08]. At a high level, a 'lossy' (or 'messy' in the terminology of [PVW08]) cryptosystem is one which has two types of public keys which specify two different modes of operation. In the normal mode, encryption is injective, while in the lossy mode, the ciphertexts generated by the encryption algorithm are independent of the plaintext. We also require that no efficient adversary can distinguish normal keys from lossy keys. In [BHY09], they also require a property

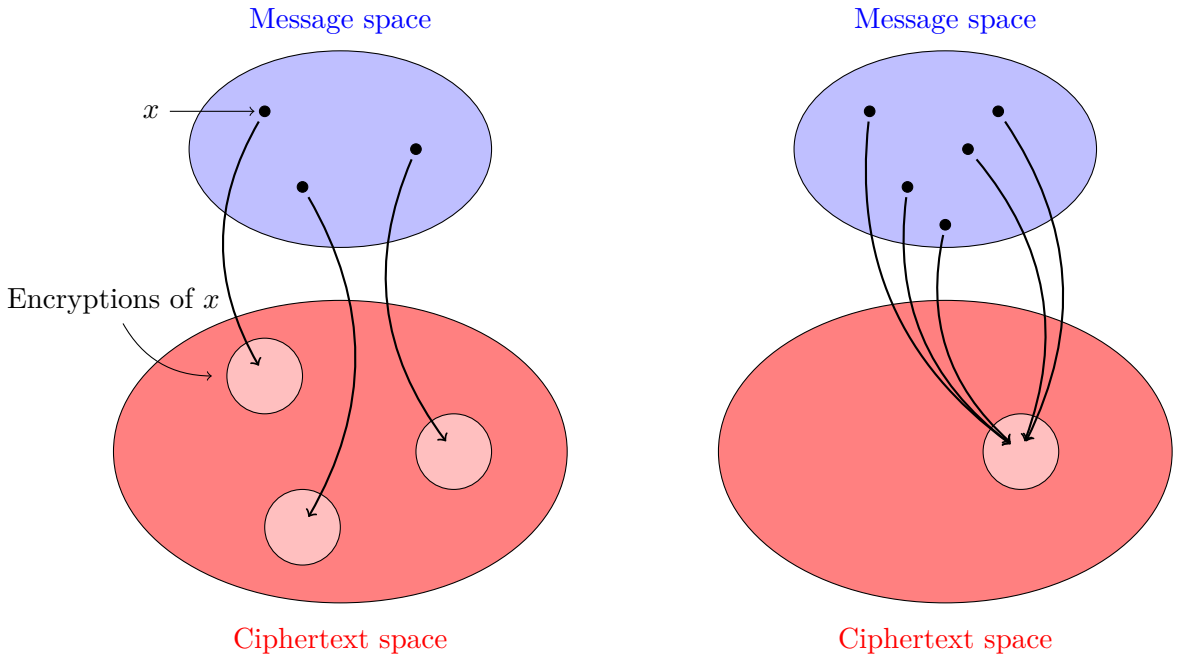


Figure 2.2: Lossy Encryption

called *openability*, which basically allows a possibly inefficient algorithm to open a ciphertext generated under a lossy key to *any* plaintext (see Figure 2.2).

Contributions [HLOV11]. In many cryptosystems, given a ciphertext c and a public-key, it is possible to re-randomize c to a new ciphertext c' such that c and c' encrypt the same plaintext but are statistically independent. In [HLOV11], with Hemenway, Libert and Ostrovsky, we formalized the notion of *statistically re-randomizable* public-key encryption [PR07, Gro04, GJJS04, CKN03]. We showed that re-randomizable encryption implies lossy encryption, as defined in [PVW08] and expanded in [BHY09].

Combining this with the result of Bellare, Hofheinz and Yilek [BHY09] showing that lossy encryption is secure against selective opening attacks, we obtained an efficient construction of an encryption scheme secure against selective opening attacks from any re-randomizable encryption scheme. This security definition ensures that if an adversary observes many ciphertexts, and may then ask for openings (*i.e.* the plaintext and the randomness used for encryption) of some of them, then unopened ciphertexts remain secure (see Appendix C for details). In addition, we showed that lossy encryption is also implied by (honest-receiver) statistically-hiding oblivious transfer and by hash proof systems [CS02] (see Chapter 4).

Finally, we also presented definitions for chosen-ciphertext security in the selective opening setting and described encryption schemes that provably satisfy these enhanced forms of security (under the DDH assumption or the Composite Residuosity assumption [Pai99]).

2.1.2 Traceable Anonymous Encryption

Several papers studied the notion of *anonymous traceable encryption* (*e.g.* [KTY07]) in which an adversary cannot determine which user's public key has been used to generate the ciphertext that it sees while a trusted third party (given some trapdoor information) is able to revoke anonymity and thus to trace back to the intended recipient. However in most of them, an

encryption scheme may contain a steganographic channel (or a covert channel) which malicious users can use to communicate illegally using ciphertexts that trace back to nobody, or even worse to some honest user.

For instance, in 2007, Kiayias, Tsiounis and Yung [KTY07] presented *group encryption*, a cryptographic primitive that provides semantic security, anonymity and a way for the *group manager* to revoke anonymity of ciphertexts. However, their construction makes use of non-interactive zero-knowledge proofs to determine whether a ciphertext is valid or not. As a consequence, an invalid ciphertext can be used to transmit some information. Above all, subliminal channels (available in the randomness) can be exploited to send some information in addition to a clean message, or even to frame an honest user.

In 2000, Sako [Sak00] proposed a novel approach to achieve bid secrecy in auction protocols. Her technique consists in expressing each bid as an encryption of a *known* message, with a key corresponding to the value of the bid. Therefore, what needs to be hidden in the ciphertext is not the message, but the key itself; the use of anonymous traceable encryption (e.g. group encryption) seems very promising for such applications (the bid itself being identified using the tracing procedure). However, one major concern in auction protocols is the problem of collusion between bidders and it is highly desirable to prevent bidders from engaging in such collaborative bidding strategies. It is worth noting that to be secure, the auction protocol must rely on a *strongly robust* encryption scheme (see [ABN10, FLPQ13]).

Contributions [IPV10]. In [IPV10], with Izabachène and Pointcheval, we introduced a new primitive which we called *mediated anonymous traceable encryption* and that provides confidentiality and anonymity while preventing malicious users to embed subliminal messages in ciphertexts.

It is relatively easy to design an anonymous encryption scheme that provides traceability or the absence of steganographic channel but the task is more challenging if one wants to achieve both simultaneously. Indeed, the existence of the tracing procedure implies that a ciphertext contains (at least implicitly) some information about the recipient, but this value can be used to transmit one bit of covert information.

In order to provide semantic security, asymmetric encryption has to be probabilistic. We introduced a *mediator* that is not provided with any secret information, but whose role —similar to the warden model introduced by Simmons [Sim83]— is to add more randomness to each ciphertext so that any hidden message is smothered. Using technigues from malleable cryptography (namely *universal re-encryption* from [GJJS04]), we proposed efficient constructions of mediated anonymous traceable encryption in the standard model, whose security relies on DDH-like assumptions.

2.2 Malleability in Signature Schemes

2.2.1 Security of Blind signatures

In his seminal paper [Cha82], Chaum proposed an RSA-based blind signature scheme that relies on the malleability of the RSA trapdoor one-way function. In Chaum's RSA-based blind signatures, the public key is (N, e) and the signer's private key is d (with, as usual, $ed = 1 \mod \phi(N)$ and the factorization of N is unknown). The signature of a message M is $x = \mathsf{RSA}_{N,e}^{-1}(H(M)) = H(M)^d \mod N$, where $H : \{0,1\}^* \to \mathbb{Z}_N$ is a public hash function.

The blind signature protocol allows a user to obtain the signature of a message without revealing it to the signer. To do so, the user picks r uniformly at random in Z_N^* and sends $\bar{M} = r^e \cdot H(M) \mod N$ to the signer; the signer computes $\bar{x} = \mathsf{RSA}_{N,e}^{-1}(\bar{M}) = \bar{M}^d \mod N$ and

returns $\bar{x}$ to the user, who extracts $x = \bar{x} \cdot r^{-1} \mod N$. The correctness of the scheme relies on the homomorphic property of the RSA trapdoor one-way function.

One can see that the signed message is perfectly hidden from the signer since the user uses a multiplicative version of the one-time pad encryption scheme in order to hide it. The signer then signs the encrypted message and the user can recover the signature on the original message since the encryption procedure and the signing procedure *commute* in some sense.

It has been observed that there is little hope of proving the security of this construction based only on the "standard" one-wayness assumption of the RSA function. The security of the scheme seems to rely on different, and probably stronger, properties of the underlying one-way function. In 2001, Bellare, Namprempre, Pointcheval and Semanko [BNPS03] introduced the notion of *one-more one-way function.* A function f is one-more one-way if it can be computed by some algorithm in polynomial time (in the input size) but for which there exists no probabilistic polynomial-time algorithm $\mathcal{A}$ with non-negligible probability to win the following game:

- $\mathcal{A}$ gets the description of f as input and has access to two oracles;
- an *inversion* oracle that given y in f's codomain returns x in f's domain such that $f(x) = y$;
- a *challenge* oracle that, each time it is invoked (it takes no inputs), returns a random challenge point from f's codomain;
- $\mathcal{A}$ wins the game if it succeeds in inverting all n points output by the challenge oracle using strictly less than n queries to the inversion oracle.

Bellare *et al.* showed how these problems lead to a proof of security for Chaum's RSA-based blind signature scheme in the random oracle model.

In 2003, Boldyreva [Bol03] proposed several variants of the BLS signature [BLS04]. The blind signature described in [Bol03] considers a bilinear structure $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, generated respectively by g_1,g_2 and $e(g_1, g_2)$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear form. The secret key is an element x picked uniformly at random in $\mathbb{Z}_p$ and the public key is $X = (X_1, X_2) = (g_1^x, g_2^x)$. The BLS signature σ of a message M is given by $H(M)^x \in \mathbb{G}_1$, where $H : \{0,1\}^* \to \mathbb{G}_1$ is a hash function (modeled by a random oracle). The verification consists in checking whether $e(H(M), X_2) = e(\sigma, g_2)$ holds. As for Chaum's RSA-based blind signature, the blind signing procedure consists for the user in picking $r \in \mathbb{Z}_q$ and sending $\bar{M} = H(M) \cdot g_1^r$ to the signer who computes $\bar{\sigma} = \bar{M}^x$. The signature is finally obtained by computing $\sigma = \bar{\sigma} \cdot X_1^{-r}$. The signed message is perfectly hidden since the user uses another variant of the one-time pad encryption scheme in $\mathbb{G}_1$ and again the user can recover the signature on the original message since the encryption procedure and the signing procedure *commute*.

Boldyreva proved that her scheme is one-more unforgeable (in the random oracle model) assuming the intractability of the *one-more (static) Diffie-Hellman* problem in $\mathbb{G}_1$ which consists in receiving $n+1$ random elements $h_0, \ldots, h_n$ from $\mathbb{G}_1$ and in returning $y_0, \ldots, y_n$ such that $y_i = h_i^x$, while asking at most n queries to the (static) Diffie-Hellman oracle $(\cdot)^x$ in $\mathbb{G}_1$.

Contributions [BMV08]. Following the approach from [PV05], with Bresson and Monnerat, we gave in [BMV08] arguments showing that, for any integer $n > 1$, solving the one-more problem with access to the inversion oracle up to n times cannot be reduced to the resolution of this problem with access to this oracle limited to $n+1$ queries. Our results apply to the class of *parameter-invariant* black-box reductions and are extended in the case of the one-more discrete logarithm problems to a class of *algebraic* black-box reductions.

These separation results apply to many computational problems used in the cryptographic literature, like the one-more RSA problem and the one-more static Diffie-Hellman problem in a bilinear setting. Due to the equivalence of the unforgeability of Chaum and Boldyreva blind signatures and the intractability of the one-more RSA problem and the one-more static Diffie-Hellman problem in a bilinear setting, our results imply that it is very unlikely that one will ever be able to prove the unforgeability of these schemes under the sole assumption of the one-wayness of their respective underlying primitive.

2.2.2 Signatures on Randomizable Ciphertexts and Applications

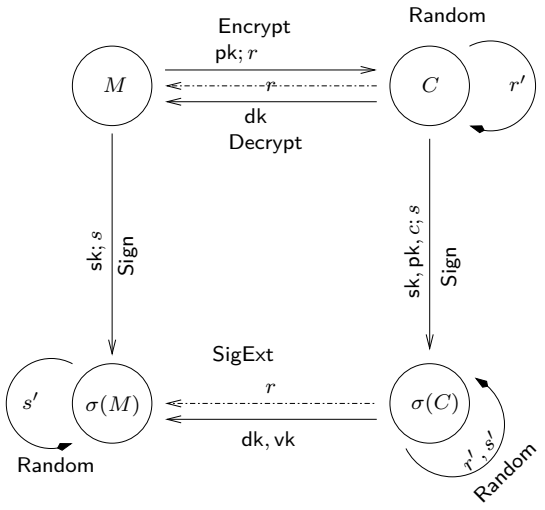
As mentioned above, homomorphic cryptographic primitives have found numerous applications. A nice side effect of homomorphic encryption is that ciphertexts can be *randomized*: given a ciphertext, anyone can—without knowing the encrypted message—produce a fresh ciphertext of the same message.

E-voting schemes make use of homomorphic encryption: users encrypt their votes under such a scheme (and add proofs and signatures), so combining the ciphertexts leads to an encryption of the election result. All signed encryptions are then made public and *verifiable*, enabling the users to check that their vote was counted, and anybody to verify the correctness of the final tally. Now, if instead of directly using a user's ciphertext, the voting center first randomizes it and proves that it did so correctly, in a non-transferable way, then users are prevented from proving the content of their vote by opening it. This deters *vote selling*, since someone buying a vote has no means to check whether the user voted as told. However, such a (non-transferable) proof of correct randomization is costly, and the randomization breaks most of the proofs of validity of the individual ciphertexts and signatures, and thus universal verifiability.

In contrast to e-voting, there are situations where encryption and signing are not performed by the same person; consider a user that encrypts a message and asks for a signature on the ciphertext. Assume now that the user can compute from this an actual signature on the message (rather than on an encryption thereof). The signature on the ciphertext could then be seen as an encrypted signature on the message, which can be decrypted by the user. This resembles the approach used in the Chaum and Boldyreva schemes described in the previous section to design blind signatures. Indeed the signer made a signature on an unknown message; but he may later recognize the signature (knowing the random coins he used) and thus break blindness (since the underlying scheme in Chaum and Boldyreva schemes are deterministic, they actually achieve the blindness property). A possible remedy are *randomizable signatures*, which allow to transform a given signature into a new one on the same message. Such signatures, a classical example being Waters signatures [Wat05] (see Section 1.4.2), do not satisfy *strong* unforgeability, which requires that it be impossible even to create a new signature on a signed message. This apparent weakness can actually be a feature, as it can be exploited to achieve *unlinkability*: the blindness property is achieved by randomizing a signature after reception.

Contributions [BFPV11, BPV12a, BFPV13]. Randomizable encryption allows anyone to transform a ciphertext into a fresh ciphertext of the same message. Analogously, a randomizable signature can be transformed into a new signature on the same message. In [BFPV11, BPV12a, BFPV13], with Blazy, Fuchsbauer and Pointcheval, we combined randomizable encryption and signatures to a new primitive called *signatures on randomizable ciphertexts* as follows: given a signature on a ciphertext, anyone, knowing neither the signing key nor the encrypted message, can randomize the ciphertext and *adapt* the signature to the fresh encryption, thus maintaining public verifiability. We also extendeded our primitive to *extractable* signatures on randomizable ciphertexts: given the decryption key, from a signature on a ciphertext one can *extract* a

signature on the encrypted plaintext (see Figure 2.3).



A message M can be encrypted using random coins r (Encrypt).
The signer can sign this ciphertext (Sign) and anyone can randomize the pair (Random).
A signature on the plaintext can be obtained using either dk (for SigExt) or the coins r (if $\sigma(C)$ has not been randomized); the result is the same as a signature of M by the signer (Sign).

Figure 2.3: (Strong) extractable signatures on randomizable ciphertexts

This primitive (whose syntactic definition is given in Figure 2.4) is related to *verifiably-encrypted-signature* schemes [BGLS03]. The latter enables a signer to make a digital signature on a message, encrypt the signature under a third party's encryption key, and produce a proof asserting that the ciphertext contains a valid signature.

In our primitive, the message is only available as an encryption and the signer does not know what message he is actually signing. It is also related to the *commuting signatures* from [Fuc11]. This primitive enables a user in a blind-signature scheme to recover a signature on the message after the signer has signed an encryption of it. As adapting a signature to a randomized encryption contradicts the standard notion of unforgeability, we introduced a weaker notion stating that no adversary can, after querying signatures on ciphertexts of its choice, output a signature on an encryption of a *new* message. This is reasonable since, due to extractability, a signature on an encrypted message can be interpreted as an encrypted signature on the message.

Moreover, exploiting the fact that the underlying encryption scheme is homomorphic, we constructed a non-interactive, receipt-free, universally verifiable e-voting scheme as follows: the user encrypts his vote, proves its validity, and sends the encryption, a signature on it, and the proof to the voting center. The latter can now randomize the ciphertext, *adapt* both the proof and the user's signature, and publish them. After the results are announced, the user can verify his signature, which convinces him that the randomized ciphertext still contains his original vote due to our notion of unforgeability; however he cannot prove to anyone what his vote was.

Using Groth-Sahai proofs and Waters signatures, we gave several instantiations of our primitive and prove them secure under classical assumptions in the standard model (see Section 3.4.1).

2.3 Proxy Re-Cryptography

2.3.1 Proxy Re-Encryption

Many papers in the literature – the first one of which being [MO97] – consider applications where data encrypted under a public key should eventually be encrypted under a different key. In *proxy encryption* schemes [Jak99, ID03], a receiver Alice allows a delegatee Bob to decrypt

A *signature scheme on randomizable ciphertexts* is an 8-tuple of polynomial-time algorithms

$$(\mathsf{Setup}, \mathsf{KeyGen}_{\mathcal{E}}, \mathsf{KeyGen}_{\mathcal{S}}, \mathsf{Encrypt}, \mathsf{Sign}, \mathsf{Verif}, \mathsf{Random}, \mathsf{SigExt}) :$$

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param for the associated encryption and signature schemes;
- $\mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param})$ generates a pair of keys, the encryption key pk and the decryption key dk;
- $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$ generates a pair of keys, the verification key vk and the signing key sk;
- $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, M; r)$ produces an encryption c of $M \in \mathcal{M}$ under pk, using the random coins r. This ciphertext is intended to be later signed under the signing key associated to the verification key vk (the argument vk can be empty if the signing algorithm is universal and does not require a ciphertext specific to the signer);
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c; s)$, produces a signature σ on a ciphertext c and a signing key sk, using the random coins $s \in \mathcal{R}_s$, or $\perp$ if c is not valid (w.r.t. pk, and possibly pk associated to sk);
- $\mathsf{Verif}(\mathsf{vk}, \mathsf{pk}, c, \sigma)$ checks whether σ is a valid signature on c, w.r.t. the public key vk. It outputs 1 if the signature is valid, and 0 otherwise (possibly because of an invalid ciphertext c, with respect to pk, and possibly vk);
- $\mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r', s')$ outputs a ciphertext c' that encrypts the same message as c under the public key pk, and a signature σ' on c'. Further inputs are a signature σ on c under vk, and random coins $r' \in \mathcal{R}_e$ and $s' \in \mathcal{R}_s$.
- $\mathsf{SigExt}(\mathsf{vk}, \mathsf{dk}, c, \sigma)$ recovers a signature on the initial plaintext m encrypted in c, valid under vk.

Figure 2.4: Signatures on Randomizable Ciphertexts

ciphertexts intended for her with the help of a proxy by providing them with shares of her private key. This requires delegatees to store an additional secret for each new delegation. Ivan and Dodis [ID03] notably present efficient proxy encryption schemes based on RSA, the Decision Diffie-Hellman problem as well as in an identity-based setting [Sha84, BF03] under bilinear-map-related assumptions.

In 2008, Blaze, Bleumer and Strauss [BBS98] proposed a cryptographic primitive called *proxy re-encryption* (PRE), in which a proxy transforms a ciphertext computed under Alice's public key into one that can be opened using Bob's secret key but where Bob only needs to store his own decryption key. A naive way for Alice to have a proxy implementing such a mechanism is to simply store her private key at the proxy: when a ciphertext arrives for Alice, the proxy decrypts it using the stored secret key and re-encrypts the plaintext using Bob's public key. The obvious problem with this strategy is that the proxy learns the plaintext and Alice's secret key.

Blaze *et al.* [BBS98] proposed the first proxy re-encryption scheme, where the plaintext and secret keys are kept hidden from the proxy. It is based on a simple modification of the ElGamal encryption scheme [Gam85]: let $(\mathbb{G}, \cdot)$ be a group of prime order p and let g be a generator of $\mathbb{G}$; Alice and Bob publish the public keys $y_A = g^a$ and $y_B = g^b$ (respectively) and keeps secret their discrete logarithms a and b. To send a message $m \in \mathbb{G}$ to Alice, a user picks uniformly

at random an integer $r \in \mathbb{Z}_p$ and transmits the pair (C_1, C_2) where $C_1 = y_A^r$ and $C_2 = m \cdot g^r$. The proxy is given the re-encryption key $b/a \mod p$ to divert ciphertexts from Alice to Bob via computing $(C_1^{b/a}, C_2) = (y_B^r, m \cdot g^r)$.

This scheme is efficient and semantically secure under the Decision Diffie-Hellman assumption in $\mathbb{G}$. It solves the above mentioned problem since the proxy is unable to learn the plaintext or secret keys a or b. Unfortunately, Blaze *et al.* pointed out an inherent limitation: the proxy key b/a also allows translating ciphertexts from Bob to Alice, which may be undesirable in some situations. They left open the problem to design a proxy re-encryption scheme which is *unidirectional*, *i.e.* where the information released to divert ciphertexts from Alice to Bob cannot be used to translate ciphertexts in the opposite direction. Another shortcoming of their scheme is that the proxy and the delegatee can collude to expose the delegator's private key a given b/a and b.

In 2005, Ateniese, Fu, Green and Hohenberger [AFGH05, AFGH06] showed how to construct *unidirectional* schemes using bilinear maps and simultaneously prevent proxies from colluding with delegatees in order to expose the delegator's long term secret. Their schemes involve two distinct encryption algorithms: *first-level* encryptions are not translatable whilst *second-level* encryptions can be re-encrypted by proxies into ciphertexts that are openable by delegatees. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a cryptographic bilinear structure of prime order p and let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Alice and Bob publish the public keys $y_A = g_1^a$ and $y_B = g_1^b$ (respectively) and keep secret their discrete logarithms a and b. To encrypt a message $m \in \mathbb{G}_T$ to Alice at the second level, a sender picks a random $r \in \mathbb{Z}_p^*$ and transmits the pair (C_1, C_2) where $C_1 = y_A^r$ and $C_2 = m \cdot e(g_1, g_2)^r$. The proxy is given the re-encryption key $g_2^{b/a}$ and can translate ciphertexts from Alice to Bob by computing $(e(C_1, g_2^{b/a}), C_2) = (e(g_1, g_2)^{br}, m \cdot e(g_1, g_2)^r)$. The decryption operations are similar to those of the ElGamal cryptosystem [Gam85].

This construction is efficient, semantically secure assuming the intractability of decisional variants of the Bilinear Diffie-Hellman problem [BF03].

These PRE schemes only ensure chosen-plaintext security, which seems definitely insufficient for many practical applications. In 2007, Canetti and Hohenberger [CH07] gave a definition of security against chosen ciphertext attacks for PRE schemes and described an efficient construction satisfying this definition. In their model, ciphertexts should remain indistinguishable even if the adversary has access to a re-encryption oracle (translating adversarially-chosen ciphertexts) and a decryption oracle (that "undoes" ciphertexts under certain rules). Their security analysis takes place in the standard model (without the random oracle heuristic [BR93]). Like the Blaze-Bleumer-Strauss scheme [BBS98], their construction is *bidirectional* and they left as an open problem to come up with a chosen-ciphertext secure unidirectional scheme.

Contributions [LV11]. In spite of these advances, in 2008, the "*holy grail for proxy re-encryption schemes – a unidirectional, key optimal, and CCA2 secure scheme – is not yet realized*" [Hoh06]. With Libert in [LV08c, LV11], we investigated this open issue.

We generalized Canetti and Hohenberger's work [CH07] and presented the first construction of chosen-ciphertext secure *unidirectional* proxy re-encryption scheme in the standard model. Our system is efficient and requires a reasonable bilinear complexity assumption. It builds on the unidirectional scheme from [AFGH06] described above. The technique used by Canetti-Hohenberger to acquire CCA-security does not directly apply to this scheme because, in a straightforward adaptation of [CH07] to [AFGH06], the validity of translated ciphertexts cannot be publicly checked. To overcome this difficulty, we needed to modify (and actually randomize) the re-encryption algorithm of Ateniese *et al.* so as to render the validity of re-encrypted ciphertexts publicly verifiable.

Whenever Alice delegates some of her rights to another party, there is always the chance that she will either need or want to revoke those rights later on. In [AFGH06], Ateniese *et al.* designed another unidirectional PRE scheme that allows for temporary delegations: that is, a scheme where re-encryption keys can only be used during a restricted time interval. We also constructed such a scheme with temporary delegation and chosen-ciphertext security.

2.3.2 Traceable Proxy Re-Encryption.

A proxy re-encryption scheme is said *non-transferable* if the proxy and a set of colluding delegatees cannot re-delegate their decryption rights [Hoh06]. The first question that comes to mind is whether transferability is really preventable since the delegatee can always decrypt and forward the plaintext. However, the difficulty in retransmitting data restricts this behavior. The security goal is therefore to prevent the delegatee and the proxy to provide another party with a secret value that can be used *offline* to decrypt the delegator's ciphertexts. Obviously, the delegatee can always send its secret key to this party, but in doing so, it assumes a security risk that is potentially injurious to itself. In the simple unidirectional system described in the previous section, colluders can unfortunately disclose $g_2^{1/a}$ which is clearly harmless to the cheating delegatee and allows for the offline opening of second level ciphertexts encrypted for the delegator. All other existing unidirectional [AFGH06] schemes are actually vulnerable to this kind of attack.

A desirable security goal is therefore to prevent a malicious proxy (or a collusion of several rogue proxies) interacting with users to take such actions. This non-transferability property has been elusive in the literature until 2008. This is not surprising since, given that proxies and delegatees can always decrypt level 2 ciphertexts by combining their secrets, they must be able to jointly compute data that allows decrypting and, once revealed to a malicious third party, ends up with a transfer of delegation. Therefore, discouraging such behaviors seems much easier than preventing them.

Contributions [LV08b]. In [LV08b], with Libert, we introduced a new notion, that we called *traceable proxy re-encryption* (TPRE), where proxies that reveal their re-encryption key to third parties can be identified by the delegator. The primitive does not preclude illegal transfers of delegation but provides a disincentive to them. Unlike prior unidirectional PRE systems, when delegators come across an illegally formed re-encryption key, they can determine its source among potentially malicious proxies. It also allows tracing delegatees and proxies that pool their secrets to disclose a pirate decryption sub-key which suffices to decipher ciphertexts originally intended for the delegator. Identifying dishonest delegatees is useful in applications such as PRE-based file storage systems [AFGH06] where there is a single proxy (i.e. the access control server) and many delegatees (i.e. end users). When a pirate decryption sub-key is disclosed in such a situation, we can find out which client broke into the access control server to generate it.

Deterring potentially harmful actions from parties that are *a priori* trustworthy may seem overburden-some: no one would elect a delegatee without having high confidence in his honesty. In these regards, the present work is somehow related to ideas from Goyal [Goy07] that aim at avoiding to place too much trust in entities (i.e. trusted authorities in identity-based encryption schemes) that must be trusted anyway. Arguably, users are less reluctant to grant their trust when abuses of delegated power are detectable and thereby discouraged.

We formalized security notions for TPRE and give efficient implementations meeting these requirements under different pairing-related assumptions. Our constructions borrow techniques from *traitor tracing* schemes [CFNP00]. We also made use of a special kind of *identity-based encryption* (IBE) system (where arbitrary strings such as email addresses [Sha84, BF03] can act as a public keys so as to avoid costly digital certificates), introduced in 2006 by Abdalla *et al.* and called *wildcard identity-based encryption* (WIBE) [ABC+11].

Our main scheme is fairly efficient, with ciphertexts of logarithmic size in the number of delegations, but the tracing system is non-black-box. Its security relies on (formerly used) mild pairing-related assumptions and the security analysis takes place in the standard model (without the random oracle heuristic [BR93]).

We also discussed how the scheme can be equipped with a black-box tracing mechanism at the expense of longer ciphertexts. The design principle is to associate re-encryption keys with codewords from a collusion-secure code [BS98]. This scheme is inspired from a WIBE-based identity-based traitor tracing scheme [ABC+11] and inherits its disadvantages: its computational overhead and the size of ciphertexts are linear in the length of the underlying code.

2.3.3 Proxy Re-Signatures

In their paper, Blaze, Bleumer and Strauss [BBS98] also introduced a cryptographic primitive where a semi-trusted proxy is provided with some information that allows turning Alice's signature on a message into Bob's signature on the same message.

As above in a naive solution, Alice – the delegator – can easily designate a proxy translating signatures computed using the secret key of Bob – the delegatee – into one that are valid w.r.t. her public key by storing her secret key at the proxy. Upon receiving Bob's signatures, the proxy can check them and re-sign the message using Alice's private key. The obvious problem with this approach is that the proxy can sign arbitrary messages on behalf of Alice. Proxy re-signatures aim at securely enabling the delegation of signatures without fully trusting the proxy. They are related to proxy signatures [MUO96, ID03] in that any PRS can be used to implement a proxy signature mechanism but the converse is not necessarily true.

In their paper [BBS98], Blaze *et al.* gave the first example of PRS where signing keys remain hidden from the proxy. The primitive was formalized in 2005 by Ateniese and Hohenberger [AH05] who pinned down useful properties that can be expected from proxy re-signature schemes:

- Unidirectionality: re-signature keys can only be used for delegation in one direction;
- Multi-usability: a message can be re-signed a polynomial number of times;
- Privacy of proxy keys: re-signature keys can be kept secret by honest proxies;
- Transparency: users may not even know that a proxy exists;
- Unlinkability: a re-signature cannot be linked to the signature from which it was generated;
- Key optimality: a user is only required to store a constant amount of secret data;
- Non-interactivity: the delegatee does not act in the delegation process;
- Non-transitivity: proxies cannot re-delegate their re-signing rights.

Blaze *et al.*'s construction is *bidirectional* (*i.e.* the proxy information allows "translating" signatures in either direction) and *multi-use* (*i.e.* the translation of signatures can be performed in sequence and multiple times by distinct proxies without requiring the intervention of signing entities). Unfortunately, Ateniese and Hohenberger [AH05] pinpointed a flaw in the latter scheme: given a signature/re-signature pair, anyone can deduce the re-signature key that has been used in the delegation (*i.e.* proxy keys are not private). Another issue in [BBS98] is that the proxy and the delegatee can collude to expose the delegator's secret.

To overcome these limitations, Ateniese and Hohenberger [AH05] proposed two constructions

based on bilinear maps. The first one is a multi-use, bidirectional extension of Boneh-Lynn-Shacham (BLS) signatures [BLS04]. Their second scheme is unidirectional (the design of such a scheme was an open problem raised in [BBS98]) but single-use. As for unidirectional proxy re-encryption schemes, it involves two different signature algorithms: *first-level* signatures can be translated by the proxy whilst *second-level* signatures (that are obtained by translating first level ones or by signing at level 2) cannot. A slightly less efficient variant was also suggested to ensure the privacy of re-signature keys kept at the proxy. The security of all schemes was analyzed in the random oracle model [BR93].

A number of applications were suggested in [AH05] to motivate the search for unidirectional systems: to provide proofs that a certain path was taken in a directed graph, to share and the convert digital certificates or to implement anonymizable signatures.

Contributions [LV08a]. Ateniese and Hohenberger left as open challenges the design of multi-use unidirectional systems and that of secure schemes in the standard security model. In [LV08a], with Libert, we provided solutions to both problems:

- we presented a simple and efficient system (built on the short signature put forth by Boneh *et al.* [BLS04]) which is secure in the random oracle model under an appropriate extension of the Diffie-Hellman assumption;
- using the elegant technique due to Waters [Wat05], the scheme is easily modified so as to achieve security in the standard model. This actually provides the first unidirectional PRS that dispenses with random oracles and thereby improves a bidirectional construction [SCWL07].

Both proposals additionally preserve the privacy of proxy keys (with an improved efficiency w.r.t. [AH05] in the case of the first one). They combine almost all of the above properties. As in prior unidirectional schemes, proxies are not completely transparent since signatures have different shapes and lengths across successive levels. The size of our signatures actually grows linearly with the number of past translations: signatures at level ℓ (*i.e.* that have been translated $\ell - i$ times if the original version was signed at level i) consist of about 2ℓ group elements. In spite of this blow-up, we retain important benefits:

- signers may tolerate a limited number (say t) of signature translations for specific messages. Then, if L distinct signature levels are permitted in the global system, users can directly sign messages at level $L - t$.
- the conversion of a ℓ^{th} level signature is indistinguishable from one generated at level $\ell + 1$ by the second signer. The original signer's identity is moreover perfectly hidden and the verifier only needs the new signer's public key.

Chapter 3

Groth-Sahai Proof System and Applications

In a zero-knowledge proof system, a prover convinces a verifier *via* an interactive protocol that a mathematical statement is true, without revealing anything else than the validity of the assertion. In 1988, Blum, Feldman, and Micali [BFM88] showed that the use of a common random string shared between the prover and the verifier permits to design a zero-knowledge proof system for all NP-languages without requiring interaction. These proofs, called non-interactive zero-knowledge, turned out to be a particularly useful tool in constructing cryptographic primitives. Unfortunately, their work (as well as subsequent results) did not yield efficient proofs. Until recently, the random oracle model was commonly used in practical instantiations which needed either non-interactive zero-knowledge proofs or non-interactive witness-indistinguishable proofs.

In 2008, Groth and Sahai [GS08] proposed a way to produce efficient and practical non-interactive zero-knowledge and non-interactive witness-indistinguishable proofs for (algebraic) statements related to groups equipped with a bilinear map. They proposed three instantiations of their system based on different (mild) computational assumptions: the subgroup decision problem (SD), the symmetric external Diffie-Hellman problem (SXDH) and the decision linear problem (DLIN). These proofs have been significantly studied in cryptography and used in a wide variety of applications in recent years, including group signature schemes, blind signatures, anonymous voting, and anonymous credentials. In this chapter, we briefly present the Groth-Sahai proof systems and several applications they found in various contexts in cryptography (*e.g.* [LV09, FPV09, BFI+10, FV10, BFPV11, BPV12a, BFPV13]).

3.1 Brief Description of Groth-Sahai Proof Systems

Groth and Sahai have introduced a methodology to build non-interactive zero-knowledge and non-interactive witness indistinguishable proofs of algebraic statements in groups equipped with a bilinear map. In this chapter, we consider only asymmetric bilinear group defined by a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, generated respectively by g_1, g_2 and $e(g_1, g_2)$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear form. To describe the proof system, we will use $\langle \cdot, \cdot \rangle$ for bilinear products between vectors of either scalars or group elements. For $\vec{a}, \vec{b} \in \mathbb{Z}_p^n$ and $(\vec{\mathcal{A}}, \vec{\mathcal{B}}) \in \mathbb{G}_1^n \times \mathbb{G}_2^n$, we define:

$$\langle \vec{a}, \vec{b} \rangle := \sum_{i=1}^{n} a_i \cdot b_i \quad \langle \vec{a}, \vec{\mathcal{A}} \rangle := \prod_{i=1}^{n} \mathcal{A}_i^{a_i} \quad \langle \vec{a}, \vec{\mathcal{B}} \rangle := \prod_{i=1}^{n} \mathcal{B}_i^{a_i} \quad \langle \vec{\mathcal{A}}, \vec{\mathcal{B}} \rangle := \prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{B}_i)$$

and for $\Gamma = (\gamma_{i,j})_{\substack{1\leq i\leq m\\1\leq j\leq n}} \in \mathbb{Z}_p^{m\times n}$

$$\Gamma\vec{\mathcal{A}} = \left(\prod_{i=1}^n \mathcal{A}_i^{\gamma_{1,i}}, \ldots, \prod_{i=1}^n \mathcal{A}_i^{\gamma_{m,i}})\right) \qquad \Gamma\vec{\mathcal{B}} = \left(\prod_{i=1}^n \mathcal{B}_i^{\gamma_{1,i}}, \ldots, \prod_{i=1}^n \mathcal{B}_i^{\gamma_{m,i}})\right).$$

The three types of statements handled by such proofs are the following:

A *pairing-product equation* over variables $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\vec{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n$ is of the form

$$\langle\vec{\mathcal{A}}, \vec{\mathcal{Y}}\rangle \cdot \langle\vec{\mathcal{X}}, \vec{\mathcal{B}}\rangle \cdot \langle\vec{\mathcal{X}}, \Gamma\vec{\mathcal{Y}}\rangle = t_T \ , \tag{3.1}$$

defined by constants $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\vec{\mathcal{B}} \in \mathbb{G}_2^m$, $\Gamma = (\gamma_{i,j})_{\substack{1\leq i\leq m\\1\leq j\leq n}} \in \mathbb{Z}_p^{m\times n}$ and $t_T \in \mathbb{G}_T$.

A *multi-scalar multiplication equation* over variables $\vec{y} \in \mathbb{Z}_p^n$ and $\vec{\mathcal{X}} \in \mathbb{G}_1^m$ is of the form

$$\langle\vec{y}, \vec{\mathcal{A}}\rangle \cdot \langle\vec{b}, \vec{\mathcal{X}}\rangle \cdot \langle\vec{y}, \Gamma\vec{\mathcal{X}}\rangle = T, \tag{3.2}$$

defined by the constants $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\vec{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m\times n}$ and $T \in \mathbb{G}_1$.

A multi-scalar multiplication equation in group $\mathbb{G}_2$ is defined analogously.

A *quadratic equation in* $\mathbb{Z}_p$ over variables $\vec{x} \in \mathbb{Z}_p^m$ and $\vec{y} \in \mathbb{Z}_p^n$ is of the form

$$\langle\vec{a}, \vec{y}\rangle + \langle\vec{x}, \vec{b}\rangle + \langle\vec{x}, \Gamma\vec{y}\rangle = t, \tag{3.3}$$

defined by the constants $\vec{a} \in \mathbb{Z}_p^n$, $\vec{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m\times n}$ and $t \in \mathbb{Z}_p$.

Groth and Sahai have detailed generic construction of the proofs π and specific instantiations under different security assumptions. We will focus on the one based on ElGamal commitments (or SXDH instantiation) since it provides the most efficient implementation and only on pairing-product equations due to space constraints.

3.1.1 Pairing-Product Equations in SXDH Instantiation

In order to generate a proof of such relations, the methodology invites us to commit to the witness vectors $\vec{\mathcal{X}}$ with randomness $\vec{R}$, and to $\vec{\mathcal{Y}}$ with $\vec{S}$ with two *double ElGamal commitments*, one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$ with respective commitment keys $\mathbf{u} \in \mathbb{G}_1^{2\times 2}$ and $\mathbf{v} \in \mathbb{G}_2^{2\times 2}$. In $\mathbb{G}_1$, a commitment $\mathbf{c}$ to a message $M \in \mathbb{G}_1$, is generated as $\mathbf{c} = (u_{1,1}^{\alpha}u_{2,1}^{\beta}, Mu_{1,2}^{\alpha}u_{2,2}^{\beta})$ (using randomness $\alpha, \beta \in \mathbb{Z}_p$). The commitment key $\mathbf{u} \in \mathbb{G}_1^{2\times 2}$ can be generated in two ways: as a a perfectly hiding commitment key $\mathbf{u} = (u_{1,1} = g, u_{1,2} = g^{\mu}, u_{2,1} = g^{\nu}, u_{2,2} = g^{\mu\nu+1})$ or as a a perfectly binding commitment key $\mathsf{ck} = (u_{1,1} = g, u_{1,2} = g^{\mu}, u_{2,1} = g^{\nu}, u_{2,2} = g^{\mu\nu})$ (using randomness $(\mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^2$). The two settings are indistinguishable under the DDH assumption in $\mathbb{G}_1$ and in the second case, the commitment is extractable (with extraction key μ) since one can retrieve the message M from $\mathbf{c}$ as $M = c_2c_1^{-\mu}$. As both commitments schemes in $\mathbb{G}_1$ and $\mathbb{G}_2$ need to be semantically secure, we will work under the SXDH assumption, so on Type III curves [GPS08].

To describe Groth-Sahai proofs, we will use abstract notations introduced in [GS08]. We will note

$$\iota_1(g_1) = (1_1, g_1), \quad \iota_2(g_2) = (1_2, g_2), \quad \iota_T(t_T) = \begin{pmatrix} 1_T & 1_T \\ 1_T & t_T \end{pmatrix}$$

and we extend linearly the notation to vectors. The symbol " $\odot$ " will denote the Hadamard product in matrices.

$$(a_{ij})_{\substack{i\in[\![1,k]\!]\\j\in[\![1,n]\!]}} \odot (b_{ij})_{\substack{i\in[\![1,k]\!]\\j\in[\![1,n]\!]}} = (a_{ij}b_{ij})_{\substack{i\in[\![1,k]\!]\\j\in[\![1,n]\!]}}$$

We will note " $\bullet$ " the distributed pairing: $\mathbb{G}_1^{n\times 2} \times \mathbb{G}_2^{n\times 2} \to \mathbb{G}_T^{2\times 2}$:

$$\vec{\mathbf{c}} \bullet \vec{\mathbf{d}} := \begin{pmatrix} \prod_{i=1}^{n} e(c_{i,1}, d_{i,1}) & \prod_{i=1}^{n} e(c_{i,1}, d_{i,2}) \\ \prod_{i=1}^{n} e(c_{i,2}, d_{i,1}) & \prod_{i=1}^{n} e(c_{i,2}, d_{i,2}) \end{pmatrix}.$$

To prove that one knows $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\vec{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n$ satisfying the pairing-product equation (3.1), a prover computes $(\vec{\mathbf{c}}, \vec{\mathbf{d}}) \in \mathbb{G}_1^{2\times m} \times \mathbb{G}_2^{2\times n}$ commitments to the witnesses $\vec{\mathcal{X}}$ and $\vec{\mathcal{Y}}$ with respective randomness $\vec{R} \in \mathbb{Z}^{2\times m}, \vec{S} \in \mathbb{Z}^{2\times n}$. It then outputs a proof $\pi = (\phi, \theta)$, together with $(\vec{\mathbf{c}}, \vec{\mathbf{d}})$, made of at most four elements in $\mathbb{G}_1$ and four in $\mathbb{G}_2$:

$$\begin{aligned} \phi = &\quad \vec{R}^\top \iota_2(\vec{\mathcal{B}}) + \vec{R}^\top \Gamma \iota_2(\vec{\mathcal{Y}}) + (\vec{R}^\top \Gamma \vec{S} - \vec{T}^\top)\mathbf{v} \\ \theta = &\quad \vec{S}^\top \iota_1(\vec{\mathcal{A}}) + \vec{S}^\top \Gamma^\top \iota_1(\vec{\mathcal{X}}) + \vec{T}\mathbf{u} \end{aligned}$$

for a random matrix $\vec{T} \in \mathbb{Z}_p^{n\times m}$, where $\vec{R}^\top, \vec{S}^\top, \vec{T}^\top$ denote the transpose of the matrix $\vec{R}, \vec{S}, \vec{T}$ respectively.

To verify the validity of $\pi = (\phi, \theta)$, a verifier checks if the equality

$$\left(\iota_1(\vec{\mathcal{A}}) \bullet \vec{\mathbf{d}}\right) \odot \left(\vec{\mathbf{c}} \bullet \iota_2(\vec{\mathcal{B}})\right) \odot \left(\vec{\mathbf{c}} \bullet \Gamma\vec{\mathbf{d}}\right) = \iota_T(t_T) \odot \left(\vec{\mathbf{u}} \bullet \phi\right) \odot \left(\theta \bullet \vec{\mathbf{v}}\right)$$

holds. Moreover, the proof and commitments are randomizable in a straightforward way.

The *Soundness* and the *Witness Indistinguishability* of such a proof directly come from the security of the commitment, and the extra randomness $\vec{T}$. Intuitively each term in the proof is here to compensate some part introduced by the randoms in the verification equation: $\vec{R}^\top \iota_2(\vec{\mathcal{B}})$ will be matched with the random part in $(\vec{\mathbf{c}} \bullet \iota_2(\vec{\mathcal{B}}))$, $\vec{S}^\top \iota_1(\vec{\mathcal{A}})$ with $(\iota_1(\vec{\mathcal{A}}) \bullet \vec{\mathbf{d}})$, $\vec{R}^\top \Gamma \iota_2(\vec{\mathcal{Y}}), \vec{S}^\top \Gamma^\top \iota_1(\vec{\mathcal{X}})$ will each annihilate the extra terms in the pairing between one of the plaintext with the commitment of the other, while $\vec{R}^\top \Gamma \vec{S}\mathbf{v}$ will remove the pairing between the two randoms. $((\vec{\mathbf{c}} \bullet \Gamma\vec{\mathbf{d}})$ can be viewed as $\vec{\mathcal{X}}_i \bullet \Gamma\vec{\mathbf{d}} \odot \vec{\mathbf{c}} \bullet \Gamma\vec{\mathcal{Y}}_i \odot \vec{R} \odot \Gamma\vec{S})$, the extra terms in $\vec{T}$ are here to randomize the proof.

Examples

1. *Proof of equality:* Let us consider an equation like $e(\mathcal{X}_1, g_2)/e(\mathcal{X}_2, g_2) = 1_T$. We commit to $\mathcal{X}_i$ in $\mathbb{G}_1$ by computing $\mathbf{c}_i = \iota_1(\mathcal{X}_i) + R_i\mathbf{u} = (u_{1,1}^{r_{1,i}} u_{2,1}^{r_{2,i}}, \mathcal{X}_i u_{1,2}^{r_{1,i}} u_{2,2}^{r_{2,i}})$. The proof is then: $\phi = \vec{R}^\top \iota_2(\mathcal{B}) - \vec{T}^\top \mathbf{v}, \theta = \vec{T}\mathbf{u}$. In this case θ does not hide the value $\vec{T}$, therefore we can only use $\pi = \phi = \vec{R}^\top \iota_2(\vec{\mathcal{B}})$. Furthermore due to the nature of ι_2, $\pi = \begin{pmatrix} 1_2 & g_2^{r_{1,1}+r_{2,1}} \\ 1_2 & g_2^{r_{1,2}+r_{2,2}} \end{pmatrix}$, and so we only need 2 group elements in $\mathbb{G}_2$ for the proof.
   The initial equation without any variable $\mathcal{Y} \in \mathbb{G}_2$ is called a linear pairing product equation (see below).

2. *Proof of a Diffie Hellman tuple:* Let us consider an equation like $e(\mathcal{X}, g_2)/e(g_1, \mathcal{Y}) = 1_T$.

   (a) The prover picks random $(r_1, r_2, s_1, s_2) \xleftarrow{\$} \mathbb{Z}_p^4$ and computes the commitments to the variables:
   $$\mathbf{c} = (u_{1,1}^{r_1} u_{2,1}^{r_2}, \mathcal{X} u_{1,2}^{r_1} u_{2,2}^{r_2}), \mathbf{d} = (v_{1,1}^{s_1} v_{2,1}^{s_2}, \mathcal{Y} v_{1,2}^{s_1} v_{2,2}^{s_2}),$$

   (b) The equation is a pairing product equation, where $\mathcal{A} = g_1^{-1}, \mathcal{B} = g_2$ and Γ is null, so the prover now picks $\vec{T} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ and computes:

$$\begin{aligned} \phi &= \vec{R}^\top \mathcal{B} - \vec{T}^\top \mathbf{v} = \begin{pmatrix} v_{1,1}^{-t_{1,1}} v_{2,1}^{-t_{2,1}} & g_2^{r_1} v_{1,2}^{-t_{1,1}} v_{2,2}^{-t_{2,1}} \\ v_{1,1}^{-t_{1,2}} v_{2,1}^{-t_{2,2}} & g_2^{r_2} v_{1,2}^{-t_{1,2}} v_{2,2}^{-t_{2,2}} \end{pmatrix}, \\ \theta &= \vec{S}^\top \mathcal{A} + \vec{T}\mathbf{u} = \begin{pmatrix} u_{1,1}^{t_{1,1}} u_{2,1}^{t_{1,2}} & g_1^{-s_1} u_{1,2}^{t_{1,1}} u_{2,2}^{t_{1,2}} \\ u_{1,1}^{t_{2,1}} u_{2,1}^{t_{2,2}} & g_1^{-s_2} u_{1,2}^{t_{2,1}} u_{2,2}^{t_{2,2}} \end{pmatrix} \end{aligned}$$

(c) To check the proof, one needs to check if:

$$\big[\iota_1(\mathcal{A}) \bullet \vec{\mathbf{d}}\,\big] + \big[\vec{\mathbf{c}} \bullet \iota_2(\mathcal{B})\big] \stackrel{?}{=} \big[\mathbf{u} \bullet \pi\big] + \big[\theta \bullet \mathbf{v}\big].$$

In other words, does $L = \big[(1_1, g_1^{-1}) \bullet (v_{1,1}^{s_1} v_{2,1}^{s_2}, Y v_{1,2}^{s_1} v_{2,2}^{s_2})\big] \odot \big[(u_{1,1}^{r_1} u_{2,1}^{r_2}, X u_{1,2}^{r_1} u_{2,2}^{r_2}) \bullet (1_2, g_2)\big]$ equal $R = \big[\mathbf{u} \bullet \pi\big] \odot \big[\theta \bullet \mathbf{v}\big]$?

$$\begin{aligned}
L &= \begin{pmatrix} 1_T & 1_T \\ e(g_1^{-1}, v_{1,1}^{s_1} v_{2,1}^{s_2}) & e(g_1^{-1}, Y v_{1,2}^{s_1} v_{2,2}^{s_2}) \end{pmatrix} \odot \begin{pmatrix} 1_T & e(u_{1,1}^{r_1} u_{2,1}^{r_2}, g_2) \\ 1_T & e(X u_{1,2}^{r_1} u_{2,2}^{r_2}, g_2) \end{pmatrix} \\
&= \begin{pmatrix} 1_T & e(u_{1,1}^{r_1} u_{2,1}^{r_2}, g_2) \\ e(g_1^{-1}, v_{1,1}^{s_1} v_{2,1}^{s_2}) & e(g_1^{-1}, v_{1,2}^{s_1} v_{2,2}^{s_2}) \cdot e(u_{1,2}^{r_1} u_{2,2}^{r_2}, g_2) \end{pmatrix} \odot \begin{pmatrix} 1_T & 1_T \\ 1_T & e(g_1^{-1}, Y).e(X, g_2) \end{pmatrix} \\
R &= \begin{pmatrix} e(u_{1,1}, v_{1,1}^{-t_{1,1}} v_{2,1}^{-t_{2,1}}) \cdot e(u_{2,1}, v_{1,1}^{-t_{1,2}} v_{2,1}^{-t_{2,2}}) & e(u_{1,1}, g_2^{r_1} v_{1,2}^{-t_{1,1}} v_{2,2}^{-t_{2,1}}) \cdot e(u_{2,1}, g_2^{r_2} v_{1,2}^{-t_{1,2}} v_{2,2}^{-t_{2,2}}) \\ e(u_{1,2}, v_{1,1}^{-t_{1,1}} v_{2,1}^{-t_{2,1}}) \cdot e(u_{2,2}, v_{1,1}^{-t_{1,2}} v_{2,1}^{-t_{2,2}}) & e(u_{1,2}, g_2^{r_1} v_{1,2}^{-t_{1,1}} v_{2,2}^{-t_{2,1}}) \cdot e(u_{2,2}, g_2^{r_2} v_{1,2}^{-t_{1,2}} v_{2,2}^{-t_{2,2}}) \end{pmatrix} \\
&\odot \begin{pmatrix} e(u_{1,1}^{t_{1,1}} u_{2,1}^{t_{1,2}}, v_{1,1}) \cdot e(u_{1,1}^{t_{2,1}} u_{2,1}^{t_{2,2}}, v_{2,1}) & e(u_{1,1}^{t_{1,1}} u_{2,1}^{t_{1,2}}, v_{1,2}) \cdot e(u_{1,1}^{t_{2,1}} u_{2,1}^{t_{2,2}}, v_{2,2}) \\ e(g_1^{-s_1} u_{1,2}^{t_{1,1}} u_{2,2}^{t_{1,2}}, v_{1,1}).e(g_1^{-s_2} u_{1,2}^{t_{2,1}} u_{2,2}^{t_{2,2}}, v_{2,1}) & e(g_1^{-s_1} u_{1,2}^{t_{1,1}} u_{2,2}^{t_{1,2}}, v_{1,2}).e(g_1^{-s_2} u_{1,2}^{t_{2,1}} u_{2,2}^{t_{2,2}}, v_{2,2}) \end{pmatrix} \\
&= \begin{pmatrix} 1_T & e(u_{1,1}^{r_1} u_{2,1}^{r_2}, g_2) \\ e(g_1^{-1}, v_{1,1}^{s_1} v_{2,1}^{s_2}) & e(g_1^{-1}, v_{1,2}^{s_1} v_{2,2}^{s_2}).e(u_{1,2}^{r_1} u_{2,2}^{r_2}, g_2) \end{pmatrix}
\end{aligned}$$

We have $L = R$ if and only if $\begin{pmatrix} 1_T & 1_T \\ 1_T & e(g_1^{-1}, Y).e(X, g_2) \end{pmatrix} = \begin{pmatrix} 1_T & 1_T \\ 1_T & 1_T \end{pmatrix}$, and therefore if and only if $e(g_1^{-1}, Y).e(X, g_2) = 1_T$.

As seen above, pairing product equations come in two forms, the *quadratic* ones, and the *linear* ones. Linear equations drastically reduce the size of the proofs. In the SXDH setting, those in $\mathbb{G}_1$ are:

$$\langle \vec{\mathcal{X}}, \vec{\mathcal{B}} \rangle = t_T \ , \langle \vec{y}, \vec{\mathcal{A}} \rangle = T \ , \qquad\qquad \langle \vec{b}, \vec{\mathcal{X}} \rangle = T \ , \langle \vec{x}, \vec{b} \rangle = t$$

Those different simplifications help to reduce the size of the proofs, and so the number of exponentiations required. For the different equations, we obtain the complexity summarized in the following table:

Assumption: SXDH	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{Z}_p$
Variables $x \in \mathbb{Z}_p, X \in \mathbb{G}_1$	2	0	0
Variables $y \in \mathbb{Z}_p, Y \in \mathbb{G}_2$	0	2	0
Pairing-Product Equation:	4	4	0
$\vec{\mathcal{A}} \cdot \vec{\mathcal{Y}} = d_T$	2	0	0
$\vec{\mathcal{X}} \cdot \vec{\mathcal{B}} = d_T$	0	2	0
Multi-Scalar Equation in $\mathbb{G}_1$:	2	4	0
$\vec{\mathcal{A}} \cdot \vec{y} = d_1$	1	0	0
$\vec{\mathcal{X}} \cdot \vec{b} = d_1$	0	0	2
Multi-Scalar Equation in $\mathbb{G}_2$:	4	2	0
$\vec{x} \cdot \vec{\mathcal{B}} = d_2$	0	1	0
$\vec{a} \cdot \vec{\mathcal{Y}} = d_2$	0	0	2
Quadratic equations in $\mathbb{Z}_p$:	2	2	0
$\vec{a} \cdot \vec{y} = d$	0	0	1
$\vec{x} \cdot \vec{b} = d$	0	0	1

3.1.2 Efficient Verification of Groth-Sahai Proofs

In the last twenty years, there has been a lot of work in cryptography in which expensive tasks are processed in batch rather than individually to achieve better efficiency. Batch cryptography was first introduced by Fiat [Fia89], who proposed an algorithm to compute several private RSA key operations (with different exponents) through one full exponentiation and several small exponentiations. Batch cryptography is particularly relevant in settings where many exponentiations need to be verified together and it seems natural to apply such techniques to the verification of Groth-Sahai proofs, which require expensive evaluations of pairings. In 1998, Bellare, Garay and Rabin [BGR98] took the first systematic look at batch verification and described several techniques for conducting batch verification of exponentiations with high confidence. They proposed three generic methods called the *random subset test*, the *small exponents test* and the *bucket test*. More recently, Ferrara, Green, Hohenberger and Pedersen [FGHP09], presented a detailed study on how to securely batch verify a set of pairing-based equations and some applications on existing signatures schemes.

In [BFI+10], with Blazy, Fuchsbauer, Izabachène, Jambert and Sibert, we proposed a way to batch the verification of several Groth-Sahai proofs. Our first goal was to consider the cases where we have to verify n similar equations, like when someone wants to verify a bunch of signatures. We followed the steps of [Fia89], [BGR98], and [FGHP09]. We decided to batch those expensive tasks all at once.

In order to do so, we used the *small exponent test* from [BGR98]. We picked small random exponents r_i, raised the ith-equation to power r_i and checked if the product of the left-hand sides of those randomized equations was equal to the product of the right ones. This induces a tiny soundness error (It was shown in [FGHP09] that it is bounded by $2^{-\ell}$, when r_i are ℓ-bits strings), but drastically improve the efficiency. We also followed simple rules, to avoid costly exponentiations in $\mathbb{G}_T$ by moving the exponent inside the pairing on the element in $\mathbb{G}_1$ when possible ([GPS08] explained that an exponentiation in $\mathbb{G}_2$ may be more costly).

1. **Move the exponent into the pairing:** $e(f_i, h_i)^{\delta_i} \to e(f_i^{\delta_i}, h_i)$
2. **Move the product into the pairing:** $\prod_{j=1}^{m} e(f_j^{\delta_j}, h_i) \to e\left(\prod_{j=1}^{m} f_j^{\delta_j}, h_i\right)$
3. **Switch two products:** $\prod_{j=1}^{m} e\left(f_j, \prod_{i=1}^{k} h_i^{\delta_{i,j}}\right) \leftrightarrow \prod_{i=1}^{k} e\left(\prod_{j=1}^{m} f_j^{\delta_{i,j}}, h_i\right)$

Here δ_i can be any exponent involved in the i-th equation, so of course the power r_i but also the exponent $\gamma_{i,k}$ associated with the quadratic pairing product, and public scalars in both multi-scalar multiplication equations and quadratic equations.

In [BFI+10], we applied these batch-verification techniques to the verification equations for Groth-Sahai proofs, and obtained some nice improvements. One can see that even for $n = 1$, our verification technique provides some good results presented in Table 3.1, page 31 (for the SXDH instantiation).

	Naive computation	Batch computation
Pairing-product equation	$5\,m + 3\,n + 16$	$m + 2\,n + 8$
Multi-scalar multiplication equation in $\mathbb{G}_1$	$8\,m + 2\,n + 14$	$\min(2\,n + 9, 2\,m + n + 7)$
Quadratic equation	$8\,m + 8\,n + 12$	$2\min(m, n) + 8$

Table 3.1: Number of pairings per verification, where n and m stand for the number of variables.

3.2 Group Signatures and E-Cash

3.2.1 Group Signatures

The *group signature* primitive, as introduced by Chaum and van Heyst in 1991 [Cv91], allows members of a group to sign messages, while hiding their identity within a population group members administered by a group manager. At the same time, it must be possible for a tracing authority holding some trapdoor information to "open" signatures and find out which group members are their originator.

Many group signatures were proposed in the nineties, the first provably coalition-resistant proposal being the famous scheme proposed by Ateniese, Camenisch, Joye and Tsudik in 2000 [ACJT00]. The last few years saw the appearance of new constructions using bilinear maps [BBS04, NSN04, FI05, DP06]. Among these, the Boneh-Boyen-Shacham scheme [BBS04] was the first one to offer signatures shorter than 200 bytes using the *Strong Diffie-Hellman assumption* [BB04b]. Its security was analyzed using random oracles [BR93] in the model of Bellare, Micciancio and Warinschi [BMW03] which captures all the security requirements of group signatures in two well-defined properties: *full-anonymity* – formalizing that an adversary not in possession of the group manager's secret key find it hard to recover the identity of the signer from its signature – and *full-traceability* – formalizing that no colluding set of group members can create signatures that cannot be opened, or signatures that cannot be traced back to some member of the coalition. This model, which assumes static groups where no new member can be introduced after the setup phase, was independently extended by Kiayias and Yung [KY04] and Bellare-Shi-Zhang [BSZ05] to a dynamic setting.

In these models (that are very close to each other), efficient pairing-based schemes were put forth by Nguyen and Safavi-Naini [NSN04], Furukawa and Imai [FI05] and, later on, by Delerablée and Pointcheval [DP06]. In dynamically growing groups, Ateniese *et al.* [ACHdM05] also proposed a construction without random oracles offering a competitive efficiency at the expense of a security resting on interactive assumptions that are not efficiently falsifiable [Nao03].

The first practical schemes to use Groth-Sahai methodology, or more precisely a similar idea, were the Boyen and Waters group signatures [BW06b] where the proofs were in fact derived from the original techniques from [GOS06b]. Another standard model proposal was put forth (and subsequently improved [BW07]) by Boyen-Waters [BW06b] in the static model from [BMW03] under more classical assumptions. Groth [Gro06] described a scheme with constant-size signatures without random oracles in the dynamic model [BSZ05] but signatures were still too long for practical use. Later on, Groth showed [Gro07] a fairly practical random-oracle-free group signature with signature length smaller than 2 kB and full anonymity in the model of [BSZ05] under an unnatural assumption.

In order to achieve full-anonymity in group signatures, the common approach of these schemes is the following: using a signing key provided by the group manager, a user produces a signature, encrypts it and adds a proof of its validity. For this method to work efficiently in the standard model, these signing keys have to be constructed carefully. In [BW07] for example, it is the group manager that constructs the entire signing key—which means that he can impersonate (*frame*) users. Groth [Gro07] achieves *non-frameability* by using *certified signatures* (defined in [BFPW07]): the user chooses a verification key which is signed by the issuer. A signature produced with the corresponding signing key together with the verification key and the issuer's signature on it can then be verified under the issuer's key.

Contributions [FPV09]. In [FPV09], with Fuchsbauer and Pointcheval, we introduced a new primitive, which we called *partially-blind certification*. A protocol allows an issuer to interactively issue a *certificate* to a user, of which parts are then only known to the user and cannot be

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear structure where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, generated respectively by g_1,g_2 and $e(g_1, g_2)$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear form. Let $h_1 \in \mathbb{G}_1$ and define the signer's key pair as $\mathsf{sk} := x \leftarrow \mathbb{Z}_p$ and $\mathsf{pk} = X := g_2^x$.

(1) User Choose $r, y_1 \leftarrow \mathbb{Z}_p$, compute and send: $R_1 := (g_1^{y_1} h_1)^r, \quad T := g_1^r$
and zero-knowledge proofs of knowledge of r and y_1 satisfying the relations.

(2) Signer Choose $s, y_2 \leftarrow \mathbb{Z}_p$ and compute $R := R_1 T^{y_2}$
(note that $R = (h_1 g_1^y)^r$ with $y := y_1 + y_2$.)

Send $\quad (S_1 := R^{\frac{1}{x+s}},\ S_2 := g_1^s,\ S_3 := g_2^s,\ S_4 := g_1^{y_2},\ S_5 := g_2^{y_2})$

(3) User Check whether $(S_1, S_2, S_3, S_4, S_5)$ is correctly formed:

$$e(S_2, g_2) \stackrel{?}{=} e(g_1, S_3) \qquad e(S_4, g_2) \stackrel{?}{=} e(g_1, S_5) \qquad e(S_1, X S_2) \stackrel{?}{=} e(R, g_2)$$

If so, compute a certificate

$$(A := S_1^{1/r},\ X := S_2,\ X' := S_3,\ Y := g_1^{y_1} S_4 = g_1^y,\ Y' := g_2^{y_1} S_5 = g_2^y)$$

Figure 3.1: Partially-Blind Certificates from [FPV09]

associated to a particular protocol execution by the issuer. The certificates are unforgeable similarly to blind signatures in that from q runs of the protocol with the issuer cannot be derived more than q valid certificates. We presented an efficient pairing-based instantiation of the primitive that is compatible with the Groth-Sahai proof system (see Figure 3.1).

To achieve non-frameability, we observed the following: it is not necessary that the user choose the verification key, as long as she can be sure that the private key contains enough entropy (namely, the value y_2 in Figure 3.1). Since the blind component of our instantiation of our primitive can serve as signing key, our construction applies immediately to build non-frameable group signatures.

The resulting scheme is less efficient than that from [Gro07]; however, it is based on a more natural assumption, while at the same time being of the same order of magnitude—especially compared to the first instantiations of fully-secure signatures in the standard model (e.g., [Gro06]). We think of the scheme as somehow being the "natural" extension of [BW07] in order to satisfy non-frameability.

3.2.2 Group Signatures with Verifier-Local Revocation.

Membership revocation has always been a critical issue in group signatures. The simplest solution is to generate a new group public key and provide unrevoked signers with a new signing key, which implies the group master to send a secret message to each individual signer as well as to broadcast a public message to verifiers. In some settings, it may not be convenient to send a new secret to signers after their inclusion in the group. In *verifier-local revocation group signatures*, originally suggested in [Bri03] and formalized in [BS04], revocation messages are only sent to verifiers (making the group public key and the signing procedure independent of which and how many members were excluded). The group manager maintains a (periodically updated)

revocation list which is used by all verifiers to perform the revocation test and make sure that signatures were not produced by a revoked member.

The revocation list contains a token for each revoked user. The verification algorithm accepts all signatures issued by unrevoked users and reveals no information about which unrevoked user issued the signature. However, if a user is revoked, his signatures are no longer accepted. It follows that signatures from a revoked member become linkable: to test that two signatures emanate from the same revoked user, one can simply verify signatures once using the revocation list before the alleged signer's revocation and once using the post-revocation revocation list . As a result, users who deliberately leave the group inevitably lose their privacy. The property of *backward unlinkability*, first introduced in [Son01] in the context of key-evolving group signatures, ensures that signatures that were generated by a revoked member *before* his revocation remain anonymous and unlinkable.

Boneh and Shacham [BS04] proposed a group signature with verifier-local revocation using bilinear maps in a model inspired from [BMW03]. In [NF05], Nakanishi and Funabiki extended Boneh-Shacham group signatures and devised a scheme providing backward unlinkability. They proved the anonymity of their construction under the Decision Bilinear Diffie-Hellman assumption [BF03]. In [NF06], the same authors suggested another backward-unlinkable scheme with shorter signatures. Until 2009, all known constructions of group signatures with verifier local revocation (with or without backward unlinkability) make use of the Fiat-Shamir paradigm [FS86] and thus rely on the random oracle methodology [BR93].

Contributions [LV09]. In [LV09], with Libert, we described a new verifier-local revocation group signatures scheme with backward unlinkability in the standard model. Extending the aforementioned constructions of group signatures to obtain verifier-local revocation with backward unlinkability was not straightforward. The approach used in [NF06], which can be traced back to Boneh-Shacham [BS04], inherently requires to use programmable random oracles, the behavior of which currently seems impossible to emulate in the standard model (even with the techniques developed in [HK08]). We adapted the approach used in [NF05] that permits traceability with backward unlinkablity without introducing additional random oracles. This technique, however, does not interact with the Groth-Sahai toolbox in a straightforward manner as it typically requires non-interactive zero-knowledge proofs for *pairing product equations*. Such non-interactive proofs are only known to be simulatable in non-interactive zero-knowledge under specific circumstances that are not met if we try to directly apply the technique of [NF05]. To address this technical difficulty, we used the same revocation mechanism as [NF05] but use a slightly stronger (but still falsifiable [Nao03]) assumption in the proof of anonymity.

3.2.3 Transferable Anonymous E-Cash

Electronic cash (E-Cash) systems allow users to withdraw electronic coins from a bank, and then to pay merchants using these coins preferably in an off-line manner, *i.e.* with no need to communicate with the bank or a trusted party during the payment. Finally, the merchant deposits the coins he has received at the bank.

An e-cash system should provide user anonymity against both the bank and the merchant during a purchase in order to emulate the perceived anonymity of regular cash. Von Solms and Naccache [vSN92] pointed out that perfect anonymity enables perfect crimes, and thus suggested *fair* e-cash, where an authority can trace coins that were acquired illegally. Necessity to fight money laundering also encourages the design of fair e-cash systems enabling a trusted party to revoke the anonymity of users, whenever needed. The participants of a *fair* e-cash system are thus the following: the system manager (that registers users within the system), the bank (issuing coins), users (that withdraw, transfer or spend coins), merchants to which coins are

spent, the double-spending detector, and a trusted authority, called tracer, that can trace coins, revoke anonymity and identify double-spenders.

Literature tries to improve the withdrawal and the spending processes. The compact e-cash system [CHL05] has given rise in 2005 to a new interest in e-cash by proposing the first e-cash system permitting a user to efficiently withdraw a wallet with T coins such that the space required to store these coins, and the complexity of the withdrawal protocol, are proportional to $\log(T)$ rather than to T. Another possibility of efficient withdrawal is also given in [AWSM07]. Another approach aiming at improving the spending phase is called divisible e-cash [EO94, CG07]. It enables a user to withdraw one coin and then to spend it at several occasions by dividing its value. However, for many applications, such as e-tickets or coupons [NHS99], transferability [OO89, OO91, CG08b] is a more desirable property. It is known that the size of coins grows linearly in the number of transfers [CP92].

Contributions [FPV09]. As mentioned above, in [FPV09], with Fuchsbauer and Pointcheval, we introduced the new primitive of *partially-blind certification*. Since in e-cash, the serial number of a coin needs to contain enough entropy to avoid collisions, but the user need not control it entirely, we were able to use our primitive to design a fair e-cash system.

In our proposal, coins are transferable while remaining constant in size. We circumvented the known impossibility results [CP92] by introducing a new method to trace double spenders: the users keep *receipts* when receiving coins instead of storing all information about transfers inside the coin. The amount of data a user has to deal with is thus proportional to the number of coins he received, rather than the path a coin took until reaching him.

Our construction is secure in the standard security model and provides the strongest possible notion of anonymity: a user remains anonymous even w.r.t. an entity issuing coins and able to *detect* double spendings. Note that in our context, the malleability of proofs is essential and it seems impossible to replace the Groth-Sahai techniques with the Fiat-Shamir heuristic [FS86] to improve efficiency at the expense of relying on the random oracle model.

3.3 Anonymous Credentials

Introduced by Chaum [Cha85] and extensively studied in the last two decades (*e.g.* [CL01, CL02b, CL02a, CL04, BCKL08, BCKL09] and references therein) anonymous credential systems enable users to authenticate themselves in a privacy-preserving manner. In such a protocol, a user can prove that an organization has supplied him with a certificate in such a way that the request for a certificate cannot be linked to any of its proofs of possession and multiple proofs involving the same credential cannot be linked to each other.

Anonymous credential systems usually combine two essential components. The first one is a protocol allowing a user to obtain a signature from an organization on a committed value (which is typically the user's private key) by sending a commitment to the signer and eventually obtaining a signature on the message without leaking useful information on the latter. The second component is a proof of knowledge of a signature on a committed value. Namely, the prover holds a pair (m, σ), reveals a commitment c to m and demonstrates his possession of σ as a valid signature on m. Camenisch and Lysyanskaya [CL01, CL02b] used groups of hidden order and Fujisaki-Okamoto commitments [FO97] to build the first practical realizations 10 years ago. Their approach was subsequently extended to groups of public order using bilinear maps [CL04, AMO08].

Until recently, all anonymous credential systems required users to engage in an interactive conversation with the verifier to convince him of their possession of a credential. While interaction can be removed using the Fiat-Shamir paradigm [FS86] and the random oracle

model [BR93], this methodology is limited to only give heuristic arguments in terms of security [GK03]. In 2008, Belenkiy, Chase, Kohlweiss and Lysyanskaya [BCKL08] relied on Groth-Sahai proof systems to design the first *non-interactive* anonymous credentials in the standard model. The protocol for obtaining a signature on a committed message still demands interaction but the proving phase, which is usually more frequently executed, consists of one message from the prover to the verifier. In order to obtain an efficient scheme, they introduced a new primitive named *P-signature* (as a shorthand for signatures with efficient Protocols). Their results were later extended into non-interactive anonymous credential schemes supporting credential delegation [BCKL09, Fuc11].

In many realistic applications, it is desirable to augment digital credentials with a number of user attributes (such as their citizenship, their birth date, their obtained degrees, ...) while allowing users to selectively disclose some of their attributes or efficiently prove properties about them without disclosing any other information. A natural approach is to extend classical anonymous credentials such as [CL01,CL04] using generalizations of the Pedersen commitment [Ped91] allowing to commit to n attributes at once in groups of hidden order. However, disclosing a single specific attribute entails to commit to $n-1$ attributes so as to prove that one attribute matches the disclosed value and committed attributes are the remaining certified ones. The drawback of this technique is that each proof has linear size in the overall number of attributes.

Camenisch and Groß [CG08a] suggested a completely different technique consisting of encoding attributes as prime numbers. Basically, users first obtain a signature on two committed messages: the first one is the user's private key and the second one consists of the product of all users' attributes. Later on, when the user wants to prove his ownership of a credential containing a certain attribute, he just has to prove that this attribute divides the second committed message. Camenisch and Groß also showed how users can prove that they hold an attribute appearing in some public attribute list and how to handle negated statements (namely, prove that a certain attribute is not contained in their attribute set). They also showed how to extend their techniques and prove the conjunction or the disjunction of simple such atomic statements. Unfortunately, their techniques cannot be applied in the setting of non-interactive anonymous credentials as they inherently rely on groups of hidden order, which makes them incompatible with the Groth-Sahai proof systems.

Contributions [ILV11]. In [ILV11], with Izabachène and Libert, we presented an anonymous credential scheme allowing to non-interactively prove the possession of a credential associated with attributes that satisfy a given predicate without leaking any further information. To this end, we extended the approach of [BCKL08] by introducing a new kind of P-signature termed *block-wise* P-signature. In a nutshell, this primitive is a P-signature allowing a user to obtain a signature on a committed vector of messages (similarly to the multi-block P-signature of [BCKL09]). Unlike [BCKL09] however, our P-signature makes it possible for the user to generate a short NIZK argument (*i.e.*, the size of which does not depend on the vector size) that serves as evidence that the signed vector satisfies a certain predicate.

Inspired by the work of Katz, Sahai, Waters [KSW08], we presented a block-wise P-signature for predicates corresponding to the zero or non-zero evaluation of inner products (and therefore disjunctions or polynomial evaluations). By combining our block-wise P-signature with the Groth-Sahai methodology [GS08] as in [BCKL08], we readily obtain an efficient non-interactive anonymous credential supporting efficient attributes. Using a very small amount of interaction, we were also able to handle conjunctions of atomic conditions and even more complex formulas such as CNF or DNF in two rounds. The non-interactivity property is unfortunately lost but our solution still decreases the number of rounds with respect to traditional interactive constructions.

3.4 Blind Signatures and Variants

3.4.1 Blind Signatures

There have been several constructions with highly interactive blind signature protocols (like [Oka06]), before Fischlin [Fis06] gave a generic construction of *round-optimal* blind signature schemes: the signing protocol consists of one message from the user to the signer and one response by the signer (this immediately implies concurrent security). The construction relies on commitment, encryption, signature schemes and generic non-interactive zero-knowledge proofs for NP-languages: the user first sends a commitment to the message to the signer who responds with a signature on the commitment. The (blind) signature is then an encryption of the commitment and the signature together with a a non-interactive zero-knowledge proof that the signature is valid on the commitment and that the committed value is the message.

Fuchsbauer et al. [Fuc09, AFG+10] have efficiently instantiated his blueprint. In [Fuc09], Fuchsbauer introduced the notion of *automorphic signatures* whose verification keys lie in the message space, messages and signatures consist of group elements only, and verification is done by evaluating a set of pairing-product equations. Among several applications, he constructed an (automorphic) blind signature in the following way: the user commits to the message, and gives the issuer a randomized message; the issuer produces a "pre-signature" from which the user takes away the randomness to recover an actual signature on the message (and not a signature on a commitment). The actual signature is then a Groth-Sahai NIWI proof of knowledge of a signature, which guarantees unlinkability to the issuing.

In 2012, Seo and Cheon [SC12] also presented a construction leading to blind signature schemes. However, it relies on a trick consisting in starting from prime-order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ and considering group elements in $\mathcal{G} = \mathbb{G}_1 \oplus \mathbb{G}_2 \oplus \mathbb{G}_3$. While their approach provides nice theoretical tools, the resulting signatures lie in $\mathcal{G}^2$ and are therefore inefficient.

Contributions [BFPV11, BFPV13]. In [BFPV11, BFPV13], with Blazy, Fuchsbauer and Pointcheval, as briefly described in the previous chapter (see Section 2.2.2), we introduced the primitive of extractable signatures on randomizable ciphertexts. We gave several instantiations of this primitive, all of which are based on very mild assumptions. Our constructions use the following building blocks, from which they inherit their security: non-interactive witness-indistinguishable Groth-Sahai proofs and the variant of Waters signatures derived from the scheme in [Wat05] described in Section 1.4.2. Since verification of Waters signatures is a statement of the language for Groth-Sahai proofs (namely, a pairing-product equation), these two building blocks combine smoothly. Our (asymmetric) Waters signature on ElGamal ciphertexts is described in Figure 3.2. It gives rise immediately to an efficient blind signature schemes using the transformation described in Section 2.2.2.

We avoided (randomizable) verifiable encryption of signatures by using signatures that are themselves randomizable. Blind signatures are thus signatures of the underlying scheme, which are much shorter than proofs of knowledge thereof. In our construction, the underlying (and thus the blind) signatures are Waters signatures [Wat05], which consist of 2 elements from a bilinear group. In comparison, the most efficient scheme by Abe et al. [AFG+10] has messages consisting of two group elements, while a signature consists of 18+16 (in $\mathbb{G}_1$ and $\mathbb{G}_2$) group elements. Furthermore, our schemes are secure under a classical assumption, while the schemes in [Fuc09, AFG+10] are based on newly introduced "q-type" assumptions. The drawback of our scheme is that, while being round-optimal, the user must send much more information to the signer during the blind signature-issuing protocol.

- $\mathsf{Setup}(1^k)$: The system generates a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, with respective generators g_1, g_2 and $g_T = e(g_1, g_2)$. For the signing part, we need an additional vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and a generator $h_1 \xleftarrow{\$} \mathbb{G}_1$.

- $\mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param})$: Choose a random scalar $\alpha \xleftarrow{\$} \mathbb{Z}_p$, which defines the secret key $\mathsf{dk} = \alpha$, and the public key as $\mathsf{pk} = U_1 = g_1^{\alpha}$.

- $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (X_1 = g_1^x, X_2 = g_2^x)$, and the secret key as $\mathsf{sk} = Y = h_1^x$.

- $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, M; r)$: For some message $M \in \{0,1\}^k$ and some random scalar $r \in \mathbb{Z}_p$, define the ciphertext as $c = (\mathcal{F}(M) \cdot U_1^r, g_1^r)$.

  We also add some proofs of validity of the ciphertext:

  - We add $C_r = \mathcal{C}(X_1^r)$ together with proof Π_r showing that $e(C_r, g_2) = e(c_2, X_2)$ and so that C_r is a commitment of X_1 raised to r: This equation is a linear pairing product equation. Therefore it requires 2 elements in $\mathbb{G}_2$.
  - A proof Π_M of knowledge of M in c, the encrypted $\mathcal{F}(M)$, which consists of the bit-commitments in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$, with proofs that each commitment is also indeed a bit commitment: $6k$ group elements in $\mathbb{G}_1$ and $6k$ in $\mathbb{G}_2$. One has to prove that c_1 is well-formed: $c_1 = (u_0 \prod_{i \in [1,k]} u_i^{M_{i2}}) \cdot U_1^{r_2}$, which is a linear multi-scalar multiplication equation in $\mathbb{G}_1$, and adds only one group element in $\mathbb{G}_1$.

  We denote by Π the global additional proof, which consists of $6k + 2$ elements in $\mathbb{G}_1$ and $6k + 4$ group elements in $\mathbb{G}_2$.

- $\mathsf{Sign}(\mathsf{sk} = Y, \mathsf{pk} = U_1, (c = (c_1, c_2), \Pi); s)$: When one wants to sign a ciphertext $c = (c_1, c_2)$, one first checks if the latter is valid using Π and produces: $\sigma = (Y c_1^s, c_2^s, U_1^s, g_1^s, g_2^s)$ if it is valid or $\perp$ in the other case.

- $\mathsf{Verif}(\mathsf{vk} = (X_1, X_2), \mathsf{pk} = U_1, c, \sigma)$: In order to check the validity of the signature, one checks if Π is valid and if the following pairing equations are verified:

  $$e(\sigma_2, g_2) = e(c_2, \sigma_5) e(\sigma_3, g_2) = e(U_1, \sigma_5) \quad e(\sigma_4, g_2) = e(g_1, \sigma_5) e(\sigma_1, g_2) = e(h_1, X_2) e(c_1, \sigma_5).$$

- $\mathsf{Random}(\mathsf{vk} = (X_1, X_2), \mathsf{pk} = U_1, (c = (c_1, c_2), \Pi), \sigma; r', s')$: This algorithm is given random scalars $r', s' \in \mathbb{Z}_p$ and publishes

  $$c' = (c_1' = c_1 \cdot U_1^{r'}, c_2' = c_2 \cdot g_1^{r'})$$
  $$\sigma' = (\sigma_1' = \sigma_1 c_1^{s'} \sigma_3^{r'} U_1^{s'r'}, \sigma_2' = \sigma_2 c_2^{s'} \sigma_4^{r'} g_1^{r's'}, \sigma_3' = \sigma_3 U_1^{s'}, \sigma_4' = \sigma_4 g_1^{s'}, \sigma_5' = \sigma_5 g_2^{s'}),$$

  together with a randomization Π' of Π.

- $\mathsf{SigExt}(\mathsf{dk}, \mathsf{vk}, \sigma)$: On a valid signature, if one knows the decryption key $\mathsf{dk} = \alpha$, one can get back a signature on M (of $F = \mathcal{F}(M)$): $\Sigma = (\Sigma_1 = \sigma_1 / \sigma_2^{\alpha}, \Sigma_2 = \sigma_4^{-1}, \Sigma_3 = \sigma_4^{-1})$. Note that one can also get the same value from the encryption random coins r, since $\Sigma_1 = \sigma_1 / U_1^r$.

Figure 3.2: Asymmetric Waters signature on ElGamal ciphertexts

3.4.2 Partially-Blind Signatures

A loophole in standard blind signatures was first identified by Abe and Okamoto [AO00]: the signer has no control at all over which messages are signed. In classical e-cash schemes, unforgeability, which restricts a user's number of coins to the number of withdrawals, was sufficient. For the case that the bank wants to include an expiration date in the message (in addition to the user-chosen serial number), Abe and Fujisaki [AF96] propose *partially blind signatures*, where the user and the signer agree on part of the message before running the blind signing protocol.

The above-mentioned scheme from [AFG+10] was extended to partially blind signature scheme in [Fuc11] and the scheme proposed by Seo and Cheon [SC12] also handles partial blindness.

Contributions [BPV12a, BFPV13]. In [BPV12a, BFPV13], with Blazy, Fuchsbauer and Pointcheval, we extended our earlier results outlined above in several directions. Instead of using an encryption scheme to blind the message to be signed, we used a *mixed commitment scheme* [DN02] in which commitments can be set up to either be perfectly binding (like encryption) or perfectly hiding.

We presented a blind signature scheme with perfect blindness, using the perfectly hiding setup of Groth-Sahai commitments [GS08]. We also extended the model of partially blind signatures to avoid prior agreement on the public part of the message between signer and user: signers can decide on its content only before sending their message in the signature-issuing protocol. Using the perfectly binding setting for the mixed commitment, we took advantage of the fact that user and signer can independently choose their inputs and consider a new context: the message to be signed is an aggregation of inputs that come from several independent sources which cannot communicate with each other.

We considered several aggregation procedures (concatenation of the inputs and addition of messages which is often used when counting votes, or aggregating sensor information). To handle the addition of messages, we reconsidered the programmable hash function used for Waters signatures over a *non-binary* alphabet, in a similar way to what Hofheinz and Kiltz [HK08] did for the binary case. Our results immediately yield Waters signatures over a non-binary alphabet, which in turn leads to a reduction in the number of public-key elements.

3.4.3 Fair Blind Signatures

Blind signatures have numerous applications including e-cash: they prevent linking withdrawals and payments made by the same customer. However, the impossibility of this linking might lead to fraud (money laundering, blackmailing, ...); some applications therefore require means to identify the resulting signature from the transcript of a signature-issuing protocol or to link a message/signature pair to user who requested it. *Fair blind signatures* were introduced by Stadler, Piveteau and Camenisch in [SPC95] to provide these means. Several fair blind signature schemes have been proposed since then [SPC95, AO01, HT07] with applications to e-cash [GT03] or e-voting [CGT06]. In [HT07], Hufschmitt and Traoré presented the first formal security model for fair blind signatures and a scheme based on bilinear maps satisfying it in the random oracle model under an interactive assumption. In 2010, Rückert and Schröder [RS10] proposed a *generic* construction of fair *partially* blind signatures [AF96].

Contributions [FV10]. In [FV10], with Fuchsbauer, we revisited this security model and proposed a stronger variant. We gave a definition of blindness analogously to [Oka06], but additionally provide tracing oracles to the adversary; in contrast to [HT07], this models *active* adversaries. We proposed a traceability notion that implies the original one and we formalized

the non-frameability notions analogously to [BSZ05], where it is the adversary's task to output a framing signature (or transcript) *and a proof*.

In order to construct the first practical fair blind signature scheme with a security reduction in the standard model, we modified Fuchsbauer's blind signature scheme [Fuc09]. We extended Fuchsbauer's automorphic signature so that it can sign three messages at once and to achieve blindness – even against adversaries provided with tracing oracles – we used Groth's technique from [Gro07] to achieve CCA-anonymous group signatures. More precisely, instead of just committing to the tracing information, we additionally encrypted it (using Kiltz' tag-based encryption scheme [Kil06]) and provide NIZK proofs of consistency with the commitments. Finally, in order to achieve the strengthened notion of non-frameability, we constructed simulation-sound NIZK proofs of knowledge of a Diffie-Hellman solution which consist of group elements only and are verified by checking a set of pairing-product equations (i.e. they are Groth-Sahai compatible).

Our fair blind signatures are Groth-Sahai compatible themselves which makes them perfectly suitable to design efficient fair e-cash systems following the approach proposed in [GT03]. In addition, our scheme is compatible with the "generic" variant of Votopia [OMA+99] proposed by Canard, Gaud and Traoré in [CGT06] (which was used during the French referendum on the European Constitution in May 2005.). Combined with a suitable mix-net (e.g. [GL07]), it provides a practical electronic voting protocol in the standard model including public verifiability, and compares favorably with other similar systems in terms of computational cost.

Chapter 4

Smooth Projective Hash Proof Systems and Applications

Smooth projective hashing was introduced by Cramer and Shoup in 2002 [CS02]. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset (language) of its domain, using an additional witness of language-membership. A projective hash family is *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. An important property that is used in all the applications of such families is that it is hard to distinguish members of the special subset from non-members. This is called the *hard subset membership property* and the primitive can be seen as special type of proof system of language-membership. In this chapter, we present several notions of SPHF and the numerous applications they found in various contexts in cryptography (*e.g.* [GL03,Kal05,ACP09,BPV12b, BBC$^+$13a,BCPV13,BBC$^+$13b]).

4.1 Definitions

In [CS98], Cramer and Shoup introduced the first practical encryption scheme that was proved IND-CCA secure in the standard security model, with security based on the Decisional Diffie Hellman (DDH) Assumption. They later presented an abstraction of this scheme based on a notion that they called "smooth projective hashing" [CS02]. Basically, *Smooth Projective Hash Functions* (SPHF) are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a set $\mathcal{S}et$ containing an NP-language $\mathcal{L}$. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where hk, the hashing key, can be seen as the private key and hp, the projection key, as the public key. On a word $C \in \mathcal{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ with the hashing key and $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w)$ with the projection key and also a witness w that $C \in \mathcal{L}$. Of course, if $W \notin \mathcal{L}$, such a witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W)$ is independent of hp. As a consequence, even knowing hp, one cannot guess $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W)$ (see Figure 4.1). Moreover, if $\mathcal{L}$ is a hard partitioned subset of $\mathcal{S}et$ (*i.e.*, it is computationally hard to distinguish a random element in $\mathcal{L}$ from a random element in $\mathcal{S}et \setminus \mathcal{L}$), the SPHF also satisfies the *pseudo-randomness* property: even for a word $W \in \mathcal{L}$, but without the knowledge of a witness w, the hash value is *computationally* indistinguishable from a random element, even knowing hp.

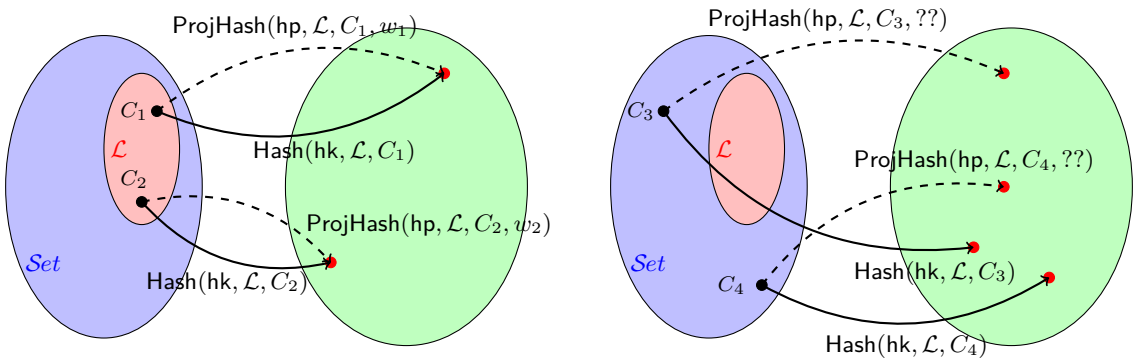


Figure 4.1: Smooth Projective Hash Functions

4.1.1 General Definition of SPHFs

Let us consider a language $\mathcal{L} \subseteq \mathcal{S}et$, and some global parameters for the SPHF, assumed to be in the common random string (CRS). The SPHF system for the language $\mathcal{L}$ is defined by four algorithms:

- $\mathsf{HashKG}(\mathcal{L})$ generates a hashing key hk for the language $\mathcal{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ derives the projection key hp, possibly depending on a word $C \in \mathcal{S}et$;
- $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ outputs the hash value of the word C from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w)$ outputs the hash value of the word C from the projection key hp, and the witness w that $C \in \mathcal{L}$.

The *correctness* of the SPHF assures that if $C \in \mathcal{L}$ with w a witness of this membership, then the two ways to compute the hash values give the same result:

$$\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w).$$

On the other hand, the security is defined through different notions that capture some ways to limit the amount of information given by $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ about the behavior of the hash function on $\mathcal{S}et \backslash \mathcal{L}$. We say the hash function family is

- is ϵ-*universal*$_1$ if for any $C \in \mathcal{S}et \backslash \mathcal{L}$ and for a randomly chosen hk, the probability of correctly guessing $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ from C and $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ is at most ϵ.
- is ϵ-*universal*$_2$ if, even knowing the value of $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C^*)$ in some $C^* \in \mathcal{S}et \backslash \mathcal{L}$ for any $C \neq C^* \in \mathcal{S}et \backslash \mathcal{L}$ and for a randomly chosen hk, the probability of correctly guessing $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ from C and $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ is at most ϵ.
- H is ϵ-*smooth* if the probability distributions of $(C, \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C), \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C))$ and $(C, \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C), H)$ are ϵ-close, where hk, C and H are chosen uniformly at random by $\mathsf{HashKG}(\mathcal{L})$, in $\mathcal{S}et \backslash \mathcal{L}$ and the co-domain of $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ respectively.

In the following, we will only consider the *smoothness* property, which guarantees that, if $C \notin \mathcal{L}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp.

We recall the definitions of SPHFs and present a classification (introduced in [BBC+13b] with Benhamouda, Blazy, Chevalier and Pointcheval) based on the dependence between words and keys. According to this classification, there are three types of SPHFs:

- the (almost) initial Cramer and Shoup [CS02] type (CS-SPHF) introduced for enhancing an IND-CPA encryption scheme to IND-CCA. This is almost[1] the initial definition of SPHF, where the projection key hp does not depend on the word C (word-independent key), but the word C cannot be chosen after having seen hp for breaking the smoothness (non-adaptive smoothness). More formally, a CS-SPHF is ε-smooth if ProjKG does not use its input C and if, for any $C \in \mathcal{S}et \backslash \mathcal{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

- the Gennaro and Lindell [GL03] type (GL-SPHF) introduced for Password-Authenticated Key Exchange (PAKE). This is a relaxation, where the projection key hp can depend on the word C (word-dependent key). More formally, a GL-SPHF is ε-smooth if, for any $C \in \mathcal{S}et \backslash \mathcal{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C);\ H \xleftarrow{\$} \Pi\}.$$

- the Katz and Vaikuntanathan [KV11] type (KV-SPHF) introduced for one-round PAKE. This is the strongest SPHF, in which the projection key hp does not depend on the word C (word-independent key) and the smoothness holds even if C depends on hp (adaptive smoothness). More formally, a KV-SPHF is ε-smooth if ProjKG does not use its input C and, for any function f onto $\mathcal{S}et \backslash \mathcal{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, f(\mathsf{hp}))\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

Remark 4.1.1 One can see that a perfectly smooth (*i.e.*, 0-smooth) CS-SPHF is also a perfectly smooth KV-SPHF, since each value H has exactly the same probability to appear, and so adaptively choosing C does not increase the above statistical distance. However, as soon as a weak word C can bias the distribution, f can exploit it.

4.1.2 Examples

1. *Proof of a Diffie Hellman tuple:* Let us consider a group $\mathbb{G}$ of order prime p with a generators g_1 and g_2 and the language $\mathcal{L} = \{(g_1^r, g_2^r), r \in \mathbb{Z}_p^*\} \subset \mathbb{G}^2 = \mathcal{S}et$ (*i.e.* the language of Diffie-Hellman tuples). In [CS02], Cramer and Shoup proposed the following SPHF for the language $\mathcal{L}$:

   - $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generators g_1 and g_2 and a collision-resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$;
   - $\mathsf{HashKG}(\mathcal{L})$ generates a hashing key $\mathsf{hk} = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$;
   - $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \perp)$ derives the projection key $\mathsf{hp} = g_1^{x_1} g_2^{x_2}$.
   - $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C = (u_1, u_2))$ outputs the hash value $H = u_1^{x_1} \cdot u_2^{x_2} \in \mathbb{G}$.
   - $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C = (g_1^r, g_2^r), w = r)$ outputs the hash value $H' = \mathsf{hp}^r \in G$.

[1] In the initial definition, the smoothness was defined for a word C randomly chosen from $\mathcal{S}et \backslash \mathcal{L}$, and not necessarily for any such word.

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a collision-resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$. The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^z)$ and outputs M. Otherwise, one outputs $\perp$.

Figure 4.2: (Labelled) Cramer-Shoup Encryption Scheme

Pseudorandomness follows from the DDH assumption and *Correctness* follows since

$$H' = \mathsf{hp}^r = (g_1^{x_1} g_2^{x_2})^r = (u_1^{x_1} u_2^{x_2}) = H$$

For 0-*smoothness*, if $\mathcal{C} \notin \mathcal{L}$ then H is unpredictable: Given $\mathsf{hp} = g_1^\alpha$, $g_2 = g_1^\beta$, $u_1 = g_1^r$ and $u_2 = g_2^s$, the hash value is $H = g^\gamma$ that satisfies:

$$\begin{pmatrix} \alpha \\ \gamma \end{pmatrix} = \begin{pmatrix} 1 & \beta \\ r & \beta s \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The determinant of this matrix is $\Delta = \beta(s-r)$, that is zero if and only if we do have a valid Diffie-Hellman tuple. Otherwise, the matrix is non-singular and from hp, γ is perfectly hidden, from an information theoretical point of view, and so is $\mathsf{Hash}(\mathsf{hk}, (u_1, u_2))$ too.

Viewing this hash proof system as a special type of designated verifier NIZK, we can instantiate the Naor-Yung construction [NY90], and we obtain a variant of the Cramer-Shoup encryption scheme [CS98] known as *Cramer-Shoup lite* that achieves non-adaptive IND-CCA security (IND-CCA1) under the DDH assumption. In order to achieve adaptive IND-CCA security (IND-CCA2), one need to reinforce the smoothness property in order to obtain a one-time simulation sound designated verifier NIZK. With a universal$_2$ hash proof system, we get the well-known Cramer-Shoup encryption scheme [CS98] (see Figure 4.2) that is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

2. *Proof of a validity of a Cramer-Shoup ciphertext:* Going one step further, one can consider SPHFs for the language of valid Cramer-Shoup ciphertexts (without labels). Let us consider a group $\mathbb{G}$ of order prime p with a generators g_1 and g_2, an encryption key $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$ for Cramer-Shoup Encryption Scheme (see Figure 4.2), a message $M \in \mathbb{G}$ and the language

$$\mathcal{L}_{\mathsf{CS}} = \{\mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r, r \in \mathbb{Z}_p^* \text{ and } \xi = \mathfrak{H}_K(\mathbf{u}, e)\} \subset \mathbb{G}^4 = \mathcal{S}et$$

A GL-SPHF *for* $\mathcal{L}_{\mathsf{CS}}$. In 2003, Gennaro and Lindell [GL03] proposed the following GL-SPHF on Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $C = (\mathbf{u} = (u_1, u_2) =$

$(g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^\eta g_2^\theta h^\mu (cd^\xi)^\nu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language $\mathcal{L}_{\mathsf{CS}}$ of the valid ciphertexts of M,

$$H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, M), C) \stackrel{\text{def}}{=} u_1^\eta u_2^\theta (e/M)^\mu v^\nu$$
$$= \mathsf{hp}^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, M), C, r) \stackrel{\text{def}}{=} H'.$$

A KV-SPHF *for* $\mathcal{L}_{\mathsf{CS}}$. In [BBC+13b], with Benhamouda, Blazy, Chevalier and Pointcheval, we gave the description of the first known KV-SPHF on labelled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5$; the associated projection key is the pair $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\mu c^\nu, \mathsf{hp}_2 = g_1^{\eta_2} d^\nu) \in \mathbb{G}^2$. Then one can compute the hash value in two different ways, for the language $\mathcal{L}_{\mathsf{CS}}$ of the valid ciphertexts of M under ek:

$$H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) \stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/M)^\mu v^\nu$$
$$= (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, M), C, r) = H'.$$

We will briefly present in Section 4.4.3 an application of this SPHF to Password-Authenticated Key Exchange (see also [BBC+13b] and Appendix I).

The rest of this chapter is devoted to the presentation of several applications of SPHF in various contexts (UC-Secure Commitment Schemes, Oblivious Signature-Based Envelopes and Blind signatures, Password-Authenticated Key Exchange, Language-based Authenticated Key Exchange, Proofs of Non-Membership and Anonymous Credentials).

4.2 UC-Secure Commitment Schemes

Commitment schemes are one of the most important tools in cryptographic protocols. This is a two-phase protocol between two parties, a committer and a receiver. In the first *commit* phase, the committer gives the receiver a digital analogue of a locked box containing a value m. In the second *opening* phase, the committer reveals m in such a way that the receiver can verify it. As in the locked box analogy, it is required that a committer cannot change the committed value (*i.e.*, he should not be able to open to a value different from the one he committed to), this is called the *binding* property. It is also required that the receiver cannot learn anything about m before the opening phase, this is simply called the *hiding* property.

The security definition for commitment schemes in the UC framework (see Section 1.5) was presented by Canetti and Fischlin [CF01]. A UC-secure commitment scheme achieves the binding and hiding properties under any concurrent composition with arbitrary protocols and it was shown, in [CF01], that it cannot be securely realized without additional assumptions. The common reference string (CRS) setting is the most widely used assumption when considering commitment schemes. In this setting, all parties have access to public information ideally drawn from some predefined distribution.

From a theoretical viewpoint, UC-secure commitments are an essential building block to construct more complex UC-secure protocols such as zero-knowledge protocols [DN02] and two-party or multi-party computations [CLOS02]. Moreover, a UC-secure commitment scheme provides *equivocability* (*i.e.*, an algorithm that knows a secret related to the CRS can generate commitments that can be opened correctly to any value) and *extractability* (*i.e.*, another algorithm that knows a secret related to the CRS can correctly extract the content of any valid commitment generated by anybody). Therefore, since their introduction, UC-secure commitments have

We have a CRS, consisting of $(p, \mathbb{G}, g_1, g_2, c, d, h, h_1, h_2, \zeta, \mathfrak{H}_K)$, where $\mathbb{G}$ is a group of prime order p with generators g_1, g_2; $c, d, h \in \mathbb{G}$ are random elements in $\mathbb{G}$ and $h_1 = {g_1}^\rho$ and $h_2 = {g_2}^\rho$ for a random $\rho \in \mathbb{Z}_p$; $\mathfrak{H}_K$ is randomly drawn from a collision-resistant hash function family $\mathcal{H}$. Intuitively $(p, \mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$ is a Cramer-Shoup encryption key, $(p, \mathbb{G}, g_1, g_2, h_1, h_2)$ is the CRS of a dual-mode encryption scheme, and $(p, \mathbb{G}, g, \zeta)$ is the CRS of a Pedersen commitment scheme (denoted Ped).
Let $G : \{0,1\}^n \to \mathbb{G}$ be an efficiently computable and invertible mapping of a binary string to the group.

The commit phase

Upon receiving a message $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ where $x \in \{0,1\}^{n-\log^2(n)}$ and $\mathsf{sid}, \mathsf{ssid} \in \{0,1\}^{\log^2(n)/4}$, party P_i works as follows:

1. P_i computes $m = G(x, \mathsf{sid}, \mathsf{ssid}, P_i, P_j) \in \mathbb{G}$.
2. P_i picks $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $C = \mathsf{CS}(m; r)$ a Cramer-Shoup encryption of m with randomness r for the public key $(p, \mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$. We will note ω the hash of the first three terms of C.
3. P_i picks $k_1 \xleftarrow{\$} \mathbb{Z}_p$, computes $c_p^1 = \mathsf{Ped}(\mathfrak{H}_K(C); k_1)$ and sends it to P_j.
4. P_j picks $R, S \xleftarrow{\$} \mathbb{Z}_p$, $\varepsilon \xleftarrow{\$} \{0,1\}^n$ and sends $c' = ({g_1}^R {g_2}^S, {h_1}^R {h_2}^S G(\varepsilon))$ to P_i.
5. P_i picks $s, k_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\alpha, \beta, \gamma, \delta) = ({g_1}^s, {g_2}^s, h^s, (cd^\omega)^s)$.
   He then computes and sends $c_p^2 = \mathsf{Ped}(\mathfrak{H}_K(\alpha, \beta, \gamma, \delta); k_2)$ to P_j.
6. P_j now opens c' by sending (R, S, ε) to P_i.
7. P_i checks if this is consistent with c' otherwise he aborts.
8. P_i now computes $z = s + \varepsilon r \ bmodp$ (where ε is interpreted as an integer), and erases r, s.
   He also opens c_p^1 by sending C, k_1 to P_j.
9. P_j verifies the consistency of c_p^1.
   If yes, he stores $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, c, \varepsilon, c_p^2)$ and outputs $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.
   He ignores any later commitment messages with the same $(\mathsf{sid}, \mathsf{ssid})$ from P_i.

The decommit phase

Upon receiving a message $(\mathsf{Reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$, P_i works as follows:

1. P_i sends $(x, \alpha, \beta, \gamma, \delta, k_2, z)$ to P_j.
2. P_j computes $m = G(x, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$, and outputs $(\mathsf{Reveal}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, x)$ if and only if c_p^2 is consistent and:

$$g_1^z = \alpha {u_1}^\varepsilon, {g_2}^z = \beta {u_2}^\varepsilon, h^z = \gamma (e/m)^\varepsilon, (cd^\omega)^z = \delta v^\varepsilon$$

Figure 4.3: Lindell's Commitment. UC-secure against adaptive corruptions with erasures

found numerous practical applications in cryptography (*e.g.* in the area of Authenticated Key Exchange [GL03, CHK+05b, ACP09, BBC+13a, BBC+13b]).

Several UC-secure commitment schemes in the CRS model have been proposed. Canetti and Fischlin [CF01] and Canetti, Lindell, Ostrovsky, and Sahai [CLOS02] proposed inefficient non-

The commit phase

5. P_i picks $s, k_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\alpha, \beta, \gamma, \delta) = ({g_1}^s, {g_2}^s, h^s, (cd^{\omega})^s)$.
   He then computes and sends $c_p^2 = \mathsf{Ped}(\underline{m}, \mathfrak{H}_K(\alpha, \beta, \gamma, \delta); k_2)$ to P_j.

Figure 4.4: Simple Patch to the Protocol from Figure 4.3

interactive schemes from general primitives. On the other hand, Damgård and Nielsen [DN02], and Camenish and Shoup [CS03] (among others) presented interactive constructions from several number-theoretic assumptions.

In 2011, Lindell [Lin11a] presented the first very efficient commitment schemes proven in the UC framework. They can be viewed as combinations of Cramer-Shoup encryption schemes and Σ-protocols. He presented two versions, one proven against static adversaries (static corruptions), while the other can also handle adaptive corruptions (see Figure 4.3).

These two schemes have commitment lengths of only 4 and 6 group elements respectively, while their total communication complexity amount to 14 and 19 group elements respectively. Their security relies on the classical Decisional Diffie-Hellman assumption in standard cryptographic groups. Fischlin, Libert and Manulis [FLM11] shortly after adapted the scheme secure against static corruptions by removing the interaction in the Σ-protocol using non-interactive Groth-Sahai proofs [GS08]. This transformation also makes the scheme secure against adaptive corruptions but at the cost of relying on the Decisional Linear assumption in symmetric bilinear groups. It thus requires the use of computationally expensive pairing computations for the receiver and can only be implemented over groups twice as large (rather than the ones that do not admit pairing computations).

Contributions [BCPV13]. In [BCPV13], with Blazy, Chevalier and Pointcheval, we detailed an inconsistency on the *binding* property of Lindell's scheme for adaptive corruptions proposed in the conference paper [Lin11a]. In the full version of his paper [Lin11b], Lindell acknowledged the mistake in the security proof and in order to correct the scheme, we proposed a simple patch to Lindell's scheme making it secure against adaptive corruptions (see Figure 4.4). Moreover, we also improved the efficiency of both Lindell's commitment schemes [Lin11a]. As mentioned above, the committer encrypts the value m (encoded as a group element) using the Cramer-Shoup encryption scheme [CS98]. In the opening phase, he simply reveals the value m and uses a Σ protocol to give an interactive proof that the message is indeed the one encrypted in the ciphertext. In Lindell's schemes, the challenge in the Σ protocol is sent to the committer using a "dual encryption scheme". Our improvement consists in noting that the receiver can in fact send this challenge directly without having to send it encrypted before.

With additional modifications of the schemes, we presented two new protocols secure under the DDH assumption in the UC framework, against static and adaptive corruptions. Both schemes requires a smaller bandwidth and less interactions than the original schemes:

- Static corruptions: the scheme requires the communication of 9 group elements and 3 scalars where Lindell's original proposal requires 10 group elements and 4 scalars. The commit phase is non-interactive and the opening phase needs 3 rounds (instead of 5 in Lindell's scheme).

- Active corruptions: the scheme requires the communication of 10 group elements and 4 scalars where Lindell's original proposal requires 12 group elements and 6 scalars. The commitment phase needs 3 rounds (instead of 5 in Lindell's scheme) and the opening phase is non-interactive.

4.3 Oblivious Signature-Based Envelopes and Blind signatures

The exchange of digital credentials is an increasingly popular approach for trust establishment in open distributed systems. In this setting, the possession of certain credentials may be considered as privacy sensitive information. One of the major problems in regulating the flow of sensitive credentials during trust establishment is the cyclic policy interdependency which occurs when a communication party is obliged to be the first to reveal a sensitive credential to the other.

Oblivious Signature-Based Envelopes (OSBE) were introduced in 2003 by Li, Du and Boneh [LDB03]. It can be viewed as a nice way to ease the the cyclic policy interdependency of several authentication protocols. Let us consider the following scenario:

- Alice is a member of an organization and possesses a certificate produced by an authority attesting she is a member of this organization.
- Bob wants to send a private message P to members of this organization.
- Due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization.

OSBE lets Bob sends an "obfuscated" version of this message P to Alice, in such a way that Alice will be able to retrieve P if and only if Alice is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization. This primitive is part of a growing field of protocols, around *automated trust negotiation*, which also include Secret Handshakes [BDS+03], Password-based Authenticated Key-Exchange [GL06], and Hidden Credentials [BHS04] (see Section 4.4). It can be used in client-server interactions where a client needs to access a resource anonymously, but with authorization (*e.g.* distribution of content in peer-to-peer networks).

In [LDB03], Li *et al.* presented three concrete OSBE schemes: RSA-OSBE, BLS-OSBE and Rabin-OSBE. The last two use identity-based encryption schemes (Boneh-Franklin [BF03] and Cocks [Coc01] schemes, respectively) and do not require interaction, while RSA-OSBE is a 2-round protocol. In [NT06], Nasserian and Tsudik proposed OSBE schemes for the ElGamal signature family (*i.e.* Schnorr [Sch91], Nyberg-Rueppel [NR93], ElGamal [Gam85] and DSA [Nat94]).

Contributions [BPV12b]. In [BPV12b], with Blazy and Pointcheval, we first clarified and increased the security requirements of an OSBE scheme. The main improvement resides in some protection for both the sender and the receiver against the organization authority. The OSBE notion echoes the idea of SPHF if we consider the language $\mathcal{L}$ defined by encryption of valid signatures, which is hard to distinguish under the security of the encryption schemes. We showed how to build, from a SPHF on this language, an OSBE scheme in the standard model with a CRS. We proved the security of our construction in regards of the security of the commitment (the ciphertext), the signature and the SPHF scheme. We then showed how to build a simple and efficient OSBE scheme relying on a classical assumption, DLin.

Our approach demonstrates that the notion of smooth projective hash functions is an efficient alternative for interactive protocols. This new way of using SPHFs indeed avoids the need of costly Groth-Sahai proofs when an interaction is inherently needed in the primitive. Our method does not add any other interaction, and so supplement smoothly those proofs.

As an illustration of our design principle, we also adapted the Blind Signature schemes proposed in [BFPV11] (and briefly described in Section 3.4.1). Our approach fits perfectly and decreases significantly the communicational complexity of the schemes (it is divided by more than three in one construction). The security is proved in a slightly weaker model and relies on a

In our blind signature protocol, we need to "prove" that a ciphertext encrypts a bit in exponent of a basis u_i where $\mathsf{ek} = (g_1, g_2)$ is an ElGamal encryption key. That is the language

$$\mathcal{L}_{\mathsf{ek},u_i} = \{C = (c_1, c_2) \in \mathbb{G}^2, \exists r \in \mathbb{Z}_p, c_1 = g_1^r \wedge c_2 \in \{g_2^r, g_2^r \cdot u_i\}\}.$$

This is thus a simple disjunction of two SPHFs:

- $\mathsf{HashKG}(\mathcal{L}_{\mathsf{ek},u_i})$: $\mathsf{hk} = ((x_1, x_2), (y_1, y_2)) \xleftarrow{\$} \mathbb{Z}_p^4$
- $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_{\mathsf{ek},u_i}, C)$: $\mathsf{hp} = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, \mathsf{hp}_\Delta = c_1^{x_1} c_2^{x_2} \cdot c_1^{y_1} (c_2/u_i)^{y_2})$
- $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}_{\mathsf{ek},u_i}, C)$: $v = c_1^{x_1} c_2^{x_2}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}_{\mathsf{ek},u_i}, C, r)$: If $c_2 = g_2^r$, $v' = \mathsf{hp}_1^r$, else (if $c_2 = g_2^r \cdot u_i$), $v' = \mathsf{hp}_\Delta / \mathsf{hp}_2^r$

Figure 4.5: GL-SPHF for the encryption of one bit

weakened security assumptions: the XDH assumption instead of the SXDH assumption and permits to use more bilinear group settings (namely, Type-II and Type-III bilinear groups [GPS08] instead of only Type-III bilinear groups for the construction presented in [BFPV11]). Our main tool is a simple GL-SPHF to prove that an ElGamal ciphertext is the encryption of one bit (see Figure 4.5). The role of this GL-SPHF is basically to replace the proof Π_M in the protocol described in Figure 3.2. The blind signature we obtained is described in Figure 4.6.

4.4 Authenticated Key Exchange

The main goal of an *Authenticated Key Exchange* (AKE) protocol is to enable two parties to establish a shared cryptographically strong key over an insecure network under the complete control of an adversary. AKE is one of the most widely used and fundamental cryptographic primitives. In order for AKE to be possible, the parties must have authentication means, *e.g.* (public or secret) cryptographic keys, short (*i.e.*, low-entropy) secret keys or *credentials* that satisfy a (public or secret) policy.

4.4.1 Password-Authenticated Key Exchange

PAKE, for *Password-Authenticated Key Exchange*, allows users to generate a strong cryptographic key based on a shared "human-memorable" (*i.e.* low-entropy) password without requiring a public-key infrastructure. In this setting, an adversary controlling all communication in the network should not be able to mount an *off-line dictionary attack*. The most famous instantiation has been proposed by Bellovin and Merritt [BM92], EKE for Encrypted Key Exchange, which simply consists of a Diffie-Hellman key exchange [DH76], where the flows are symmetrically encrypted under the shared password. Overall, the equivalent of 2 group elements have to be sent.

A first formal security model was proposed by Bellare, Pointcheval and Rogaway [BPR00] (the BPR model), to deal with off-line dictionary attacks. It essentially says that the best attack should be the on-line exhaustive search, consisting in trying all the passwords by successive executions of the protocol with the server. Several variants of EKE with BPR-security proofs have been proposed in the ideal-cipher model or the random-oracle model [Poi12].

Katz, Ostrovsky and Yung [KOY01] proposed the first practical scheme, provably secure in

- $\mathsf{BSSetup}(1^k)$, where k is the security parameter, generates a pairing-friendly system

$$(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$$

and an ElGamal encryption key $\mathsf{ek} = u \in \mathbb{G}_1$. It also chooses at random $h_1 \in \mathbb{G}_1$ and generators $\vec{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}_1^\ell$ for the Waters function. It outputs the global $\mathsf{param} = (p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, \mathsf{ek}, h_1, \vec{u})$;

- $\mathsf{BSKeyGen}(\mathsf{param})$ picks at random $x \in \mathbb{Z}_p$, sets $\mathsf{sk} = h_1^x$ and computes the verification key $\mathsf{vk} = (g_1^x, g_2^x)$ (note that the two elements, in $\mathbb{G}_1$ and $\mathbb{G}_2$ will be needed);

- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on M

  - $\mathcal{U}$ computes the bit-per-bit encryption of M by encrypting $u_i^{M_i}$ in

$$b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i}; r_i),$$

together with the encryption of vk_1^r in $c = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}_1^r; s)$ where $r = \sum r_i$. $\mathcal{U}$ thus sends $c = (u^{s_1}, g_1^s \mathsf{vk}^r)$ and the $b_i = (u^{r_i}, g_1^{r_i} u_i^{M_i})$;

  - On input of these ciphertexts, the algorithm $\mathcal{S}$ computes the corresponding SPHF, considering the language $\mathcal{L}$ of valid ciphertexts. This is the conjunction of the several languages:

    1. the one checking that each b_i encrypts a bit (see Figure 4.5)
    2. the second one considers (d_1, c_1, c_2) and check if (c_1, c_2) encrypts d_2 such that (d_1, d_2) is a Diffie Hellman pair in basis (u, vk_1)

    This induces a projection key composed of $3\ell + 2$ elements in $\mathbb{G}_1$.

  - $\mathcal{S}$ then computes the corresponding Hash-value v, extracts $K = \mathcal{KDF}(v) \in \mathbb{Z}_p$, generates the blinded signature $(\sigma_1'' = h_1^x \delta^s, \sigma_2' = (g_1^s, g_2^s))$ and sends $(\mathsf{hp}, Q = \sigma_1'' \times g_1^K, \sigma_2')$;

  - Upon receiving $(\mathsf{hp}, Q, \sigma_2')$, using its witnesses and hp, $\mathcal{U}$ computes the ProjHash-value v', extracts $K' = \mathcal{KDF}(v')$ and unmasks $\sigma_1'' = Q \times g^{-K'}$. Thanks to the knowledge of r, it can compute $\sigma_1' = \sigma_1'' \times (\sigma_{2,1}')^{-r}$. Note that if $v' = v$, then $\sigma_1' = h_1^x \mathcal{F}(M)^s$, which together with $\sigma_2' = (g_1^s, g_2^s)$ is a valid Waters signature on M. It can thereafter re-randomize the final signature.

- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$, checks whether $e(\sigma_1, g_2) = e(h_1, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2}) \wedge e(\sigma_{2,1}, g_2) = e(g_1, \sigma_{2,2})$.

Figure 4.6: Improved Blind Signature Scheme using GL-SPHF

the standard model under the DDH assumption. This is a 3-flow protocol, with the client sending 5 group elements plus a verification key and a signature, for a one-time signature scheme, and the server sending 5 group elements. It has been generalized by Gennaro and Lindell [GL03] who proposed a general framework to design PAKE in the CRS model using smooth projective hash functions. This approach was applied to the UC framework by Canetti, Halevi, Katz, Lindell, and MacKenzie [CHK+05b], and improved by Abdalla, Chevalier and Pointcheval [ACP09].

4.4.2 Language-based Authenticated Key Exchange

The concept of *Secret Handshakes* has been introduced in 2003 by Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS+03] (see also [JL09, AKB07]). It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded. In case of failure, the players do not learn any information about the other party's affiliation.

More recently, *Credential-Authenticated Key Exchange* (CAKE) was presented by Camenisch, Casati, Groß and Shoup [CCGS10]. In this primitive, a common key is established if and only if a specific relation is satisfied between credentials hold by the two players. This primitive includes variants of PAKE and Secret Handshakes, and so-called Verifier-based PAKE.

Contributions [BBC+13a]. In [BBC+13a], with Benhamouda, Blazy, Chevalier and Pointcheval, we proposed a new primitive that encompasses most of the previous notions of authenticated key exchange. It is closely related to CAKE and we called it LAKE, for *Language-Authenticated Key-Exchange*, since parties establish a common key if and only if they hold credentials that belong to specific (and possibly independent) languages.

In order to define the security of this primitive, we used the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provided an ideal functionality for LAKE and gave efficient realizations of the new primitive (for a large family of languages) secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions. Our realizations rely on the description of smooth projective hash functions for new interesting languages defined by linear pairing product equations on committed values (see [BBC+13a] and Appendix H).

4.4.3 One-Round Password-Authenticated Key Exchange

The ultimate step for PAKE has been achieved by Katz and Vaikuntanathan in 2011 [KV11]. They proposed a *practical* one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own passwords only. More precisely, each flow just consists of an IND-CCA ciphertext of the password and an SPHF projection key for the correctness of the partner's ciphertext (the word is the ciphertext and the witness consists of the random coins of the encryption).

Because of the simultaneous flows, one flow cannot explicitly depend on the partner's flow, which makes impossible the use of the Gennaro and Lindell SPHF (GL-SPHF), in which the projection key depends on the word (the ciphertext here). On the other hand, the adversary can wait for the player to send his flow first, and then adapt its message, which requires stronger security notions than the initial Cramer and Shoup SPHF (CS-SPHF), in which the smoothness does not hold anymore if the word is generated after having seen the projection key.

Katz and Vaikuntanathan did not manage to construct a KV-SPHF for an efficient IND-CCA encryption scheme. Instead, they suggested to use the Naor and Yung approach [NY90], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [Sah99]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [GS08] proofs in bilinear groups and the linear encryption [BBS04] which leads to a PAKE secure under the DLin assumption with a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, the two players have to send 70 group elements each.

- Players U and U' both use a (labelled) Cramer-Shoup encryption key $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$ and $\mathcal{G} : \{0,1\}^n \to \mathbb{G}$ an efficiently computable and invertible mapping of a binary string to the group.
- U, with password pw, chooses $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$,
  computes $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu})$, sets $\ell = (U, U', \mathsf{hp})$,
  and generates $C = (\mathbf{u} = (g_1^r, g_2^r), e = \mathcal{G}(\mathsf{pw}) \cdot h^r, v = (cd^{\xi})^r)$ with r a random scalar in $\mathbb{Z}_p$ and $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
  U sends $\mathsf{hp} \in \mathbb{G}^2$ and $C \in \mathbb{G}^4$ to U';
- Upon receiving $\mathsf{hp}' = (\mathsf{hp}'_1, \mathsf{hp}'_2) \in \mathbb{G}^2$ and $C' = (\mathbf{u}' = (u'_1, u'_2), e', v') \in \mathbb{G}^4$ from U', U sets $\ell' = (U', U, \mathsf{hp}')$ and $\xi' = \mathfrak{H}_K(\ell', \mathbf{u}', e')$ and computes
$$\mathsf{sk}_U = {u'_1}^{(\eta_1 + \xi' \eta_2)} {u'_2}^{\theta} (e'/\mathcal{G}(\mathsf{pw}))^{\mu} {v'}^{\nu} \cdot (\mathsf{hp}'_1 {\mathsf{hp}'_2}^{\xi})^r.$$

Figure 4.7: One-Round PAKE based on DDH

More recent results on SS-NIZK proofs or IND-CCA encryption schemes, in the discrete logarithm setting, improved on that: Libert and Yung [LY12] proposed a more efficient SS-NIZK proof of plaintext equality in the Naor-Yung-type cryptosystem with ElGamal-like encryption. The proof can be reduced from 60 to 22 group elements and the communication complexity of the resulting PAKE is decreased to 32 group elements per user. Jutla and Roy [JR12] proposed relatively-sound NIZK proofs as an efficient alternative to SS-NIZK proofs to build new publicly-verifiable IND-CCA encryption schemes. They can then decrease the PAKE communication complexity to 20 group elements per user. In any case, one can remark that all one-round PAKE schemes require pairing computations.

Contributions [BBC$^+$13b]. In [BBC$^+$13b], with Benhamouda, Blazy, Chevalier and Pointcheval, we described the instantiation of KV-SPHF on Cramer-Shoup ciphertexts given in Section 4.1.2, and thus the first KV-SPHF on an efficient IND-CCA encryption scheme. We thereafter used it within the above KV framework for one-round PAKE [KV11], in the BPR security model. Our scheme(described in Figure 4.7) just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key).

We also presented the first GL-SPHFs/KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. Those SPHFs are thus quite efficient. They lead to two applications. First, our new KV-SPHFs enable several efficient instantiations of one-round *Language-Authenticated Key-Exchange* (LAKE) protocols [BBC$^+$13a]. Our one-round PAKE scheme is actually a particular case of a more general one-round LAKE scheme, for which we provided a BPR-like security model and a security proof. Second, thanks to a new GL-SPHF, we improved on the *blind signature* scheme presented in [BPV12b] and briefly described in Section 4.3, from $5\ell + 6$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$ to $3\ell + 7$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$, for an ℓ-bit message to be blindly signed with a Waters signature [Wat05] (see [BBC$^+$13b] and Appendix I for further details).

4.5 Proofs of Non-Membership and Anonymous Credentials

In cryptography, when designing privacy-sensitive applications, the use of commitments and corresponding zero-knowledge proofs is often indispensable. A prover chooses a message m and then commits to it. He keeps the message secret and publishes the commitment. He later needs to prove that m belongs to a finite set $\mathcal{L}$ or that m does *not* belong to $\mathcal{L}$, but cannot

reveal anything about m. An important instance of this problem consists in showing that the committed value lies in a given finite set (*e.g.* in e-auctions or e-voting protocols, a bidder or voter has to prove that his secret bid or vote is chosen from a list of candidates, see [CCs08] and references therein). However one usually wants to demonstrate more complex properties about committed values. For instance in anonymous credentials systems and privacy-preserving authenticated identification or key exchange protocols, a participant must usually prove the possession of a credential issued by an authority (without revealing it).

For the latter primitives, it is often necessary to prove combination of simple statements about several credentials issued by the authority (OR, AND, and NOT connectives) [CG08a, ILV11]. For instance, a crucial requirement is that credentials issued can be later revoked. In principle, revocation lists can be used for anonymous credentials by having the user to prove in zero-knowledge that his credential is *not* contained in the list. For a finite set $\mathcal{L}$ with no additional structure, the most efficient combination of commitment and zero-knowledge proof was recently proposed by Bayer and Groth [BG13]. The interactive proof system is quite efficient: it has $O(\log(\#\mathcal{L}))$ communication and computational complexity and significantly improves the previous proposals with $O(\sqrt{\#\mathcal{L}})$ complexity [Pen11]. It can be made non-interactive in the random oracle model by using the Fiat-Shamir heuristic but their elegant technique does not generalize readily to prove the non-membership for arbitrary languages.

There exist efficient membership proofs for families of very large sets $\mathcal{L}$ equipped with an "algebraic structure" (e.g. the set of valid message/digital signatures pairs for a given public key whose cardinal is exponential in the security parameter). Most of them also admit efficient non-membership proof systems. However, up to now there is no generic construction and these zero-knowledge proofs of non-membership of committed values require specific security analysis.

A concrete setting for these non-membership proofs was introduced in 2009 by Kiayias and Zhou [KZ09] as *zero-knowledge proofs with witness elimination*. This primitive enables to prove that a committed message m belongs to a set $\mathcal{L}$ (with a witness w) in such a way that the verifier accepts the interaction only if w does not belong to a set determined by a public relation Q and some *private* input w' of the verifier. The verifier does not learn anything about w (except that $m \in \mathcal{L}$ and $(w, w') \notin Q$) and the prover does not learn anything about w'. The primitive can obviously be used to handle revocation lists. It was motivated in [KZ09] by privacy-preserving identification schemes when a user wishes to authenticate himself to a verifier while preserving his anonymity and the verifier makes sure the prover does not match the identity of a suspect user that is tracked by the authorities (without leaking any information about the suspect identity).

Contributions [BCV14]. In [BCV14], with Blazy and Chevalier, we presented an efficient non-interactive technique to prove (in zero-knowledge) that a committed message does not belong to a set $\mathcal{L}$. The proof is generic and relies on a proof of membership to $\mathcal{L}$ with specific mild properties. In particular, it is independent of the size of $\mathcal{L}$ and if there exists an efficient proof of membership for committed values, one gets readily an efficient proof of non-membership. Instantiated with a combination of *smooth projective hash functions* and Groth-Sahai proof system, we obtained very efficient realization for non-interactive proof of non-membership of committed values.

In [BCV14], we showed that the original proposal of zero-knowledge proofs with witness elimination from [KZ09] is flawed and that a dishonest prover can actually make a verifier accept a proof for a any message $m \in L$ even if $(w, w') \in Q$. In particular, in the suspect tracking scenario, a dishonest prover can identify himself even if he is on the suspect list. We explained how to apply our proof of non-membership to fix it. We obtained a proof system that achieves the security goal and is more efficient than the original (insecure) solution.

Eventually, we presented applications of our proof of non-membership to other settings such as anonymous credentials and privacy-preserving authenticated key exchange. In particular, our technique allows to remove the interactivity in the anonymous credentials with efficient attributes we proposed in [ILV11] with Izabachène and Libert (and briefly described in Section 3.3) and to enlarge the set of languages usable in LAKE [BBC$^+$13a, BBC$^+$13b].

Chapter 5

Conclusion and Perspectives

Malleable Cryptography

In the context of malleable cryptography, the primary open problem is to improve the efficiency of the existing fully homomorphic encryption schemes, to the extent that it is possible while achieving provable security under plausible assumptions [Gen09, vDGHV10, BV11a, BV11b]. These schemes have already given rise to dozens of papers presenting interesting applications which are unfortunately not practical for most of them. The techniques developed for fully-homomorphic encryption enable to develop new primitives that were believed to be unachievable by most cryptographers, namely multi-linear maps [GGH13a] and indistinguishability obfuscation [GGH$^+$13b]. These theoretical advances are very promising but making them practical is a major open problem.

In proxy re-cryptography, it would be interesting to see if multi-level unidirectional proxy re-encryption and proxy re-signature schemes have efficient realizations under more classical intractability assumptions. A perhaps more challenging task would be to find out realizations – if they exist at all – of such proxy re-signatures where the size of signatures and the verification cost do not grow linearly with the number of translations.

Groth-Sahai Proof System and Applications

Since 2008, there have been several papers that extend or improve the Groth-Sahai proof system in different directions. The batch verification technique described in Section 3.1.2 reduced the computational cost of the verification of the proofs using batch techniques, at the cost of trading perfect soundness for statistical soundness. In [Seo12], Seo gave another map for verifying proofs in the symmetric setting (DLIN instantiation) which reduces the computational cost of the verification of the proofs. The papers [GSW10, EHK$^+$13] presented other assumptions on which Groth-Sahai proofs can be based (by proposing an algebraic framework for Diffie-Hellman Assumptions). Finally [EG14] presented several additional improvements in the SXDH setting (e.g. replacement of some commitments with ElGamal encryptions, effiency improvement for the prover by letting him pick his own common reference string ...). It is therefore interesting to see if one can take advantage of these new results in order to improve the efficiency of our protocols.

Implementing systems based on Groth-Sahai proof systems turns out to be very challenging, since the resulting protocols are significantly more complex than standard crypto primitives. It would be very useful design a tool for the automatic generation of sound and efficient Groth-Sahai proofs for various algebraic languages. Such a *compiler* was successfully designed for classical Σ-protocols in [ABB$^+$10].

Smooth Projective Hash Proof Systems and Applications

In the suite of works briefly presented, we showed that in addition to classical applications the notion of smooth projective hash functions can be useful to design various interactive protocols. We developed smooth projective hash proof systems on new (algebraic) languages with efficient implementations that are of independent interest. In particular, our methodology enables us to design among the most (if not the most) efficient UC-secure commitment scheme, blind signature scheme, password authenticated key exchange schemes and anonymous credentials. Our works have already found applications in other settings [ABB+13, BP13a, BP13b] but revisiting popular constructions of privacy-preserving protocols relying on the Groth-Sahai methodology (*e.g.* [BCKL08, BCKL09]) using smooth projective hash functions are perceived to be an interesting research goal.

The construction of our SPHF has been limited to discrete-logarithm or pairing type assumptions. Recently, Blazy, Chevalier, Ducas and Pan [BCDP13] constructed an exact SPHF from lattice based assumption (namely LWE and SIS) and used it to design UC-secure commitment scheme, password authenticated key exchange and language-based authenticated key exchange schemes following our work. It seems interesting to develop similar tools for other popular cryptographic settings (and notably group of composite order) and more generally to propose a unified theory and automatic tools to design (and validate) smooth projective hash functions for various languages in these different frameworks.

Part II

Personal publications

Papers marked with an asterisk are provided in Appendix

Books

1. D. Vergnaud, *Exercices et problèmes de cryptographie*, **(Textbook in french - Foreword by Jacques Stern)** Dunod, Paris, Sciences Sup, 2012.

2. F. Amoroso and D. Vergnaud, *Minorations de la hauteur d'un nombre algébrique*, **Dottorato Di Ricerca In Matematica, Edizioni Plus**, Università di Pisa, 2004.

Books edited

1. D. Pointcheval and D. Vergnaud, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa*, **LNCS 8469**, Springer, Marrakesh, Morocco, May 2014.

2. M. Abdalla, D. Pointcheval, P.-A. Fouque and D. Vergnaud, *ACNS 2009: 7th International Conference on Applied Cryptography and Network Security*, **LNCS 5536**, Springer, Paris-Rocquencourt, France, June 2009.

Refereed journal papers

*1. O. Blazy, G. Fuchsbauer, D. Pointcheval and D. Vergnaud, *Short blind signatures*, **Journal of Computer Security** 21(5): 627-661 (2013)

2. É. Domenjoud, D. Jamet, D. Vergnaud and L. Vuillon, *Enumeration formula for* $(2, n)$*-cubes in discrete planes*, **Discrete Applied Mathematics** 160(15): 2158-2171 (2012)

*3. B. Libert and D. Vergnaud, *Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption*, **IEEE Transactions on Information Theory**, 57(3): 1786-1802 (2011)

4. B. Libert and D. Vergnaud, *Towards Practical Black-Box Accountable Authority IBE: Weak Black-Box Traceability With Short Ciphertexts and Private Keys*, **IEEE Transactions on Information Theory** 57(10): 7189-7204 (2011)

5. F. Laguillaumie and D. Vergnaud, *Time-selective convertible undeniable signatures with short conversion receipts*, **Information Sciences** 180(12): 2458-2475 (2010)

6. D. Vergnaud, *New Extensions of Pairing-Based Signatures into Universal (Multi) Designated Verifier Signatures*, **International Journal of Foundations of Computer Science** 20(1): 109-133 (2009)

7. D. Vergnaud, *Mesures d'indépendance linéaire de carrés de périodes et quasi-périodes de courbes elliptiques*, **Journal of Number Theory** 129(6): 1212-1233 (2009)

8. F. Laguillaumie and D. Vergnaud, *Multi-designated verifiers signatures: anonymity without encryption*, **Information Processing Letters** 102(2-3): 127-132 (2007)

9. F. Laguillaumie, J. Traoré and D. Vergnaud, *Universal forgery on Sekhar's signature scheme with message recovery*, **International Journal of Computer Mathematics** 81(12): 1493-1495 (2004)

Refereed international conference papers

1. Y. Dodis, D. Pointcheval, S. Ruhault, D. Vergnaud and D. Wichs *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*, **2013 ACM Conference on Computer and Communications Security, CCS 2013** (A.-R. Sadeghi, V. D. Gligor & M. Yung eds.), ACM, 2013, 647-658

*2. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval and D. Vergnaud, *New Techniques for SPHFs and Efficient One-Round PAKE Protocols* **Advances in Cryptology - Crypto 2013** (R. Canetti & J. Garay eds.) Springer, Lect. Notes Comput. Sci., vol. 8042, 2013, p. 449-475

*3. O. Blazy, C. Chevalier, D. Pointcheval and D. Vergnaud, *Analysis and Improvement of Lindell's UC-Secure Commitment Schemes*, **Applied Cryptography and Network Security, 11th International Conference, ACNS 2013** (R. Safavi-Naini & M. Locasto eds.) Springer, Lect. Notes Comput. Sci., vol. 7954, 2013, p. 534-551

4. P.-A. Fouque, D. Vergnaud and J.-C. Zapalowicz *Time/Memory/Data Tradeoffs for Variants of the RSA Problem*, **19th Annual International Computing and Combinatorics Conference, COCOON 2013** (D. Du & G. Zhang eds.) Springer, Lect. Notes Comput. Sci., 2013, vol. 7936, 2013, p. 651-662

*5. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval and D. Vergnaud, *Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages*, **16th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2013** (K. Kurosawa & G. Hanaoka eds.) Springer, Lect. Notes Comput. Sci., vol. 7778, 2013, p. 272-291

6. O. Blazy, D. Pointcheval and D. Vergnaud, *Compact Round-Optimal Partially-Blind Signatures*, **8th Conference on Security and Cryptography for Networks, SCN 2012** (I. Visconti & R. de Prisco eds.) Springer, Lect. Notes Comput. Sci., vol. 7495, 2012, p. 95-112

7. A. Guillevic and D. Vergnaud *Genus 2 Hyperelliptic Curve Families with Explicit Jacobian Order Evaluation and Pairing-Friendly Constructions*, **Pairing-Based Cryptography - Pairing 2012 - 5th International Conference** (M. Abdalla & T. Lange eds.) Springer, Lect. Notes Comput. Sci., vol. 7708, 2013, p. 234-253

8. A. Bauer, D. Vergnaud and J.-C. Zapalowicz *Inferring Sequences Produced by Nonlinear Pseudorandom Number Generators Using Coppersmith's Methods*, **15th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2012** (M. Fischlin, J. Buchman & M. Manulis eds.) Springer, Lect. Notes Comput. Sci., vol. 7293, 2012, p. 609-626

*9. O. Blazy, D. Pointcheval and D. Vergnaud, *Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions*, **9th Theory of Cryptography Conference, TCC 2012** (R. Cramer, ed.) Springer, Lect. Notes Comput. Sci., vol. 7194, 2012, p. 94-111

10. M. Izabachène, B. Libert and D. Vergnaud *Block-wise P-Signatures and Non-Interactive Anonymous Credentials with Efficient Attributes*, **Cryptography and Coding, 13th IMA International Conference** (L. Chen ed.) Springer, Lect. Notes Comput. Sci., vol. 7089, 2011, p. 431-450

*11. B. Hemenway, B. Libert, R. Ostrovsky and D. Vergnaud *Lossy Encryption: Constructions from General Assumptions and Efficient Selective Opening Chosen Ciphertext Security*, **Advances in Cryptology - Asiacrypt 2011** (D. H. Lee & H. Wang eds.) Springer, Lect. Notes Comput. Sci., vol. 7073, 2011, p. 70-88

12. D. Vergnaud *Efficient and Secure Generalized Pattern Matching via Fast Fourier Transform*, **Progress in Cryptology - Africacrypt 2011** (A. Nitaj & D. Pointcheval eds.) Springer, Lect. Notes Comput. Sci., vol. 6737, 2011, p. 41-58

13. O. Blazy, G. Fuchsbauer, D. Pointcheval and D. Vergnaud, *Signatures on Randomizable Ciphertexts*, **14th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2011** (D. Catalano, N. Fazio, R. Gennaro & A. Nicolosi eds.) Springer, Lect. Notes Comput. Sci., vol. 6571, 2011, p. 403-422

14. M. Izabachène, D. Pointcheval and D. Vergnaud *Mediated Traceable Anonymous Encryption*, **First International Conference on Cryptology and Information Security in Latin America, Latincrypt'2010** (M. Abdalla & P. S. L. M. Barreto eds.) Springer, Lect. Notes Comput. Sci. vol. 6212, 2010, p. 40-60

15. M. Joye, M. Tibouchi and D. Vergnaud *HuffŠs Model for Elliptic Curves*, **Algorithmic Number Theory, 9th International Symposium, ANTS-IX** (G. Hanrot, F. Morain & E. Thomé eds.) Springer, Lect. Notes Comput. Sci. vol. 6197, 2010, p. 234-250

16. A. Bauer, J.-S. Coron, D. Naccache, M. Tibouchi and D. Vergnaud *On The Broadcast and Validity-Checking Security of PKCS #1 v1.5 Encryption*, **Applied Cryptography and Network Security, 8th International Conference, ACNS 2010** (J. Zhou & M. Yung eds.) Springer, Lect. Notes Comput. Sci. vol. 6123, 2010, p. 1-18

17. O. Blazy, G. Fuchsbauer, M. Izabachène, A. Jambert, H. Sibert and D. Vergnaud *Batch Groth-Sahai*, **Applied Cryptography and Network Security, 8th International Conference, ACNS 2010** (J. Zhou & M. Yung eds.) Springer, Lect. Notes Comput. Sci. vol. 6123, 2010, p. 218-235

*18. G. Fuchsbauer and D. Vergnaud *Fair Blind Signatures without Random Oracles*, **Progress in Cryptology - Africacrypt 2010** (D. Bernstein & T. Lange eds.) Springer, Lect. Notes Comput. Sci. vol. 6055, 2010, p. 16-33

19. L. Dallot and D. Vergnaud *Provably Secure Code-Based Threshold Ring Signatures*, **Cryptography and Coding, 12th IMA International Conference** (M. G. Parker ed.) Springer, Lect. Notes Comput. Sci. vol. 5921, 2009, p. 222-235

20. G. Fuchsbauer, D. Pointcheval and D. Vergnaud *Transferable Constant-Size Fair E-Cash*, **International Conference on Cryptology And Network Security, CANS 2009** (J. A. Garay & A. Miyaji eds.) Springer, Lect. Notes Comput. Sci. vol. 5888, 2009, p. 226-247

*21. B. Libert and D. Vergnaud *Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model*, **International Conference on Cryptology And Network Security, CANS 2009** (J. A. Garay, A. Miyaji & A. Otsuka eds. eds.) Springer, Lect. Notes Comput. Sci. vol. 5888, 2009, p. 498-517

22. S. Canard, C. Delerablée, A. Gouget, E. Hufschmitt, F. Laguillaumie, H. Sibert, J. Traoré and D. Vergnaud *Fair E-cash: Be Compact, Spend Faster*, **Information Security, 12th**

International Conference, ISC 2009 (P. Samaranti, M. Yung, F. Martinelli & C. A. Ardagna eds.) Springer, Lect. Notes Comput. Sci. vol. 5735, 2009, p. 294-309

23. B. Libert and D. Vergnaud *Adaptive-ID Secure Revocable Identity-Based Encryption*, **Topics in cryptology - CT-RSA 2009** (M. Fischlin ed.) Springer, Lect. Notes Comput. Sci. vol. 5473, 2009, p. 1-15

24. B. Libert and D. Vergnaud *Towards Black-Box Accountable Authority IBE with Short Ciphertexts and Private Keys*, **12th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2009** (S. Jarecki & G. Tsudik eds.) Springer, Lect. Notes Comput. Sci. vol. 5443, 2009, p. 235-255.

*25. B. Libert and D. Vergnaud *Multi-Use Unidirectional Proxy Re-Signatures*, **2008 ACM Conference on Computer and Communications Security, CCS 2008** (P. Ning, P. F. Syverson & S. Jha eds.) ACM, 2008, p. 511-520

26. B. Libert and D. Vergnaud *Tracing Malicious Proxies in Proxy Re-Encryption*, **2nd International Conference on Pairing-based Cryptography - Pairing 2008** (S. Galbraith & K. Paterson eds.) Springer, Lect. Notes Comput. Sci. vol. 5209, 2008, p. 332-353.

27. E. Bresson, J. Monnerat and D. Vergnaud *Separation Results on the "One-More" Computational Problems*, **Topics in cryptology - CT-RSA 2008** (T. Malkin ed.) Springer, Lect. Notes Comput. Sci. vol. 4964, 2008, p. 71-87.

28. B. Libert and D. Vergnaud *Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption*, **11th International Conference on Practice and Theory in Public-Key Cryptography, PKC 2008** (R. Cramer ed.) Springer, Lect. Notes Comput. Sci. vol. 4939, 2008, p. 360-379

29. G. Castagnos and D. Vergnaud *Trapdoor Permutation Polynomials of $\mathbb{Z}/n\mathbb{Z}$ and Public Key Cryptosystems*, **Information Security, 10th International Conference, ISC 2007** (J. A. Garay, A. K. Lenstra, M. Mambo & R. Peralta, eds.) Springer, Lect. Notes Comput. Sci. vol. 4779, 2007, p. 333-350.

30. F. Laguillaumie and D. Vergnaud *On the Soundness of Restricted Universal Designated Signatures and Dedicated Signatures*, **Information Security, 10th International Conference, ISC 2007** (J. A. Garay, A. K. Lenstra, M. Mambo & R. Peralta, eds.) Springer, Lect. Notes Comput. Sci. vol. 4779, 2007, p. 175-188.

31. P.-L. Cayrel, A. Otmani and D. Vergnaud *On Kabatianskii-Krouk-Smeets Signatures*, **International Workshop on the Arithmetic of Finite Fields, WAIFI 2007** (C. Carlet & B. Sunar, eds.) Springer, Lect. Notes Comput. Sci. vol. 4547, 2007, p. 237-251.

32. L. El Aimani and D. Vergnaud *Gradually Convertible Undeniable Signatures*, **Applied Cryptography and Network Security, 5th International Conference, ACNS 2007** (J. Katz & M. Yung, eds.) Springer, Lect. Notes Comput. Sci. vol. 4521, 2007, p. 478-496.

33. D. Vergnaud *New Extensions of Pairing-Based Signatures into Universal Designated Verifier Signatures*, **33rd International Colloquium on Automata, Languages and Programming, ICALP 2006** (M. Bugliesi, B. Preneel, V. Sassone & I. Wegener, eds.) Springer, Lect. Notes Comput. Sci. vol. 4052, 2006, p. 58-69.

34. D. Vergnaud *RSA-Based Secret Handshakes*, **International Workshop on Coding and Cryptography, WCC 2005** (Ø. Ytrehus, ed.) Springer, Lect. Notes Comput. Sci. vol. 3969, 2006, p. 252-274.

35. F. Laguillaumie and D. Vergnaud *Short Undeniable Signatures Without Random Oracles: the Missing Link*, **Progress in Cryptology - Indocrypt 2005** (S. Maitra, C. E. Veni Madhavan & R. Venkatesan, eds.) Springer, Lect. Notes Comput. Sci. vol. 3797, 2005, p. 283-296.

36. P. Paillier and D. Vergnaud *Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log*, **Advances in Cryptology - Asiacrypt 2005** (B. Roy, ed.) Springer, Lect. Notes Comput. Sci. vol. 3788, 2005, p. 1-20.

37. F. Laguillaumie, P. Paillier and D. Vergnaud *Universally Convertible Directed Signatures*, **Advances in Cryptology - Asiacrypt 2005** (B. Roy, ed.) Springer, Lect. Notes Comput. Sci. vol. 3788, 2005, p. 682-701.

38. F. Laguillaumie and D. Vergnaud *Time-Selective Convertible Undeniable Signatures*, **Topics in cryptology - CT-RSA 2005** (A. Menezes, ed) Springer Lect. Notes Comput. Sci. vol. 3376, 2005, p. 154-171.

39. F. Laguillaumie and D. Vergnaud *Designated Verifiers Signature: Anonymity and Efficient Construction from any Bilinear Map*, **Fourth International Conference, SCN 2004** (C. Blundo & S. Cimato, eds) Springer Lect. Notes Comput. Sci. vol. 3352, 2005, p. 107-121.

40. F. Laguillaumie and D. Vergnaud *Multi-Designated Verifiers Signature Schemes*, **Sixth International Conference, ICICS 2004** (J. Lopez, S. Qing & E. Okamoto, eds.) Springer Lect. Notes Comput. Sci. vol. 3269, 2004, p. 495-507

Part III

Appendix: Articles

Malleable Cryptography

Appendix A:

Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption
IEEE IT 2011
Benoît Libert and Damien Vergnaud
This article proposes the first construction of unidirectional proxy re-encryption scheme with chosen-ciphertext security in the standard model (i.e. without relying on the random oracle idealization). The construction is efficient and requires a reasonable complexity assumption in bilinear map groups. It ensures security according to a relaxed definition of chosen-ciphertext introduced by Canetti, Krawczyk and Nielsen.

Appendix B:

Multi-Use Unidirectional Proxy Re-Signatures, ACM CCS 2008
Benoît Libert and Damien Vergnaud
This article provides the first multi-hop unidirectional proxy re-signature schemes that satisfy the requirements of the Ateniese-Hohenberger security model. The first scheme is secure in the random oracle model and it readily extends into a secure construction in the standard model. Both schemes are computationally efficient but require newly defined Diffie-Hellman-like assumptions in bilinear groups.

Appendix C:

Lossy Encryption: Constructions from General Assumptions and Efficient Selective Opening Chosen Ciphertext Security, Asiacrypt 2011
Brett Hemenway, Benoît Libert, Rafail Ostrovsky and Damien Vergnaud
This article proposes new and general constructions of lossy encryption schemes and of cryptosystems secure against selective opening adversaries. It shows that every re-randomizable encryption scheme gives rise to efficient encryptions secure against a selective opening adversary and that statistically-hiding 2-round Oblivious Transfer implies Lossy Encryption and so do smooth hash proof systems. It then presents selective opening secure commitments and encryptions from the Decisional Diffie-Hellman, Decisional Composite Residuosity *and* Quadratic Residuosity *assumptions. In an indistinguishability-based model of chosen-ciphertext selective opening security, it provides secure schemes featuring short ciphertexts under standard number theoretic assumptions and in a simulation-based definition of chosen-ciphertext selective opening security, it provides secure schemes against non-adaptive adversaries.*

Appendix A

Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption

IEEE IT 2011
[LV11] with B. Libert

Abstract : *In 1998, Blaze, Bleumer, and Strauss proposed a cryptographic primitive called* proxy re-encryption*, in which a proxy transforms – without seeing the corresponding plaintext – a ciphertext computed under Alice's public key into one that can be opened using Bob's secret key. Recently, an appropriate definition of chosen-ciphertext security and a construction fitting this model were put forth by Canetti and Hohenberger. Their system is* bidirectional*: the information released to divert ciphertexts from Alice to Bob can also be used to translate ciphertexts in the opposite direction. In this paper, we present the first construction of* unidirectional *proxy re-encryption scheme with chosen-ciphertext security in the standard model (i.e. without relying on the random oracle idealization), which solves a problem left open at CCS'07. Our construction is efficient and requires a reasonable complexity assumption in bilinear map groups. Like the Canetti-Hohenberger scheme, it ensures security according to a relaxed definition of chosen-ciphertext introduced by Canetti, Krawczyk and Nielsen.*

A.1 Introduction

The concept of proxy re-encryption (PRE) dates back to the work of Blaze, Bleumer, and Strauss in 1998 [BBS98]. The goal of such systems is to securely enable the re-encryption of ciphertexts from one key to another, without relying on trusted parties. Recently, Canetti and Hohenberger [CH07] described a construction of proxy re-encryption providing chosen-ciphertext security according to an appropriate definition of the latter notion for PRE systems. Their construction is *bidirectional*: the information to translate ciphertexts from Alice to Bob can also be used to translate from Bob to Alice. This paper answers the question of how to secure unidirectional proxy re-encryption schemes against chosen-ciphertext attacks – at least in the sense of a natural extension of the Canetti-Hohenberger definition to the unidirectional case – while keeping them efficient.

Background. In a PRE scheme, a proxy is given some information which allows turning a ciphertext encrypted under a given public key into one that is encrypted under a different key.

A naive way for Alice to have a proxy implementing such a mechanism is to simply store her private key at the proxy: when a ciphertext arrives for Alice, the proxy decrypts it using the stored secret key and re-encrypts the plaintext using Bob's public key. The obvious problem with this strategy is that the proxy learns the plaintext and Alice's secret key.

In 1998, Blaze, Bleumer and Strauss [BBS98] (whose work is sometimes dubbed BBS) proposed the first proxy re-encryption scheme, where the plaintext and secret keys are kept hidden from the proxy. It is based on a simple modification of the ElGamal encryption scheme [Gam85]: let $(\mathbb{G}, \cdot)$ be a group of prime order p and let g be a generator of $\mathbb{G}$; Alice and Bob publish the public keys $X = g^x$ and $Y = g^y$ (respectively) and keeps secret their discrete logarithms x and y. To send a message $m \in \mathbb{G}$ to Alice, a user picks uniformly at random an integer $r \in \mathbb{Z}_p$ and transmits the pair (C_1, C_2) where $C_1 = X^r$ and $C_2 = m \cdot g^r$. The proxy is given the re-encryption key $y/x \mod p$ to divert ciphertexts from Alice to Bob via computing $(C_1^{y/x}, C_2) = (Y^r, m \cdot g^r)$.

This scheme is efficient and semantically secure under the Decision Diffie-Hellman assumption in $\mathbb{G}$. It solves the above mentioned problem since the proxy is unable to learn the plaintext or secret keys x or y. Unfortunately, Blaze *et al.* pointed out an inherent limitation: the proxy key y/x also allows translating ciphertexts from Bob to Alice, which may be undesirable in some situations. They left open the problem to design a proxy re-encryption method without this restriction. Another shortcoming of their scheme is that the proxy and the delegatee can collude to expose the delegator's private key x given y/x and y.

In 2005, Ateniese, Fu, Green and Hohenberger [AFGH06] showed the first examples of *unidirectional* proxy re-encryption schemes based on bilinear maps. Moreover, they obtained the *master key security* property in that the proxy is unable to collude with delegatees in order to expose the delegator's secret. The constructions [AFGH06] are also efficient, semantically secure assuming the intractability of decisional variants of the Bilinear Diffie-Hellman problem [BF03].

These PRE schemes only ensure chosen-plaintext security, which seems definitely insufficient for many practical applications. Very recently, Canetti and Hohenberger [CH07] gave a definition of security against chosen ciphertext attacks for PRE schemes and described an efficient construction satisfying this definition. In their model, ciphertexts should remain indistinguishable even if the adversary has access to a re-encryption oracle (translating adversarially-chosen ciphertexts) and a decryption oracle (that "undoes" ciphertexts under certain rules). Their security analysis takes place in the standard model (without the random oracle heuristic [BR93]). Like the BBS scheme [BBS98], their construction is *bidirectional* and they left as an open problem to come up with a chosen-ciphertext secure unidirectional scheme.

Related Work. Many papers in the literature – the first one of which being [MO97] – consider applications where data encrypted under a public key pk_A should eventually be encrypted under a different key pk_B. In proxy encryption schemes [Jak99, ID03], a receiver Alice allows a delegatee Bob to decrypt ciphertexts intended to her with the help of a proxy by providing them with shares of her private key. This requires delegatees to store an additional secret for each new delegation. Dodis and Ivan [ID03] notably present efficient proxy encryption schemes based on RSA, the Decision Diffie-Hellman problem as well as in an identity-based setting [Sha84, BF03] under bilinear-map-related assumptions.

Proxy re-encryption schemes are a special kind of proxy encryption schemes where delegatees only need to store their own decryption key. They are generally implemented in a very specific mathematical setting and find practical applications in secure e-mail forwarding or distributed storage systems (e.g. [AFGH06]).

From a theoretical point of view, the first positive obfuscation result for a complex cryptographic functionality was recently presented by Hohenberger, Rothblum, shelat and Vaikuntanathan [HRsV07]: they proved the existence of an efficient program obfuscator for a family of circuits implementing re-encryption.

In [GA07], Green and Ateniese studied the problem of identity-based PRE and proposed a unidirectional scheme that can reach chosen-ciphertext security. Their security results are presented only in the random oracle model. Besides, the recipient of a re-encrypted ciphertext needs to know who the original receiver was in order to decrypt a re-encryption.

Our contribution. In spite of the recent advances, the "*holy grail for proxy re-encryption schemes – a unidirectional, key optimal, and CCA2 secure scheme – is not yet realized*" [Hoh06]. This paper aims at investigating this open issue.

We generalize Canetti and Hohenberger's work [CH07] and present the first construction of chosen-ciphertext secure *unidirectional* proxy re-encryption scheme in the standard model. Our system is efficient and requires a reasonable bilinear complexity assumption. It builds on the unidirectional scheme from [AFGH06] briefly recalled at the beginning of section A.3. The technique used by Canetti-Hohenberger to acquire CCA-security does not directly apply to the latter scheme because, in a straightforward adaptation of [CH07] to [AFGH06], the validity of translated ciphertexts cannot be publicly checked. To overcome this difficulty, we need to modify (and actually randomize) the re-encryption algorithm of Ateniese *et al.* so as to render the validity of re-encrypted ciphertexts publicly verifiable.

Whenever Alice delegates some of her rights to another party, there is always the chance that she will either need or want to revoke those rights later on. In [AFGH06], Ateniese *et al.* designed another unidirectional PRE scheme that allows for temporary delegations: that is, a scheme where re-encryption keys can only be used during a restricted time interval. We construct such a scheme with temporary delegation and chosen-ciphertext security.

The paper is organized as follows: we recall the concept of unidirectional proxy re-encryption and its security model in section A.2.1. We review the properties of bilinear maps and the intractability assumption that our scheme relies on in section A.2.2. Section A.3 describes the new scheme, gives the intuition behind its construction and a security proof. Section A.4 finally shows an adaptation with temporary delegation.

A.2 Preliminaries

A.2.1 Model and security notions

This section first recalls the syntactic definition of unidirectional proxy re-encryption suggested by Ateniese *et al.* [AFGH06]. We then consider an appropriate definition of chosen-ciphertext security for unidirectional PRE schemes which is directly inferred from the one given by Canetti and Hohenberger [CH07] in the bidirectional case. Like [CH07], we consider security in the *replayable* CCA sense [CKN03] where a harmless mauling of the challenge ciphertext is tolerated.

Definition A.2.1 A (single hop) unidirectional PRE scheme consists of a tuple of algorithms $(\mathsf{Global\text{-}setup}, \mathsf{Keygen}, \mathsf{ReKeygen}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{ReEnc}, \mathsf{Dec}_1, \mathsf{Dec}_2)$:

- $\mathsf{Global\text{-}setup}(\lambda) \rightarrow \mathsf{par}$: this algorithm is run by a trusted party that, on input of a security parameter λ, produces a set par of common public parameters to be used by all parties in the scheme.

- $\mathsf{Keygen}(\lambda, \mathsf{par}) \rightarrow (sk, pk)$: on input of common public parameters par and a security parameter λ, all parties use this randomized algorithm to generate a private/public key pair (sk, pk).

- $\mathsf{ReKeygen}(\mathsf{par}, sk_i, pk_j) \rightarrow R_{ij}$: given public parameters par, user i's private key sk_i and user j's public key pk_j, this (possibly randomized) algorithm outputs a key R_{ij} that allows

re-encrypting second level ciphertexts intended to i into first level ciphertexts encrypted for j.

- $\mathsf{Enc}_1(\mathsf{par}, pk, m) \rightarrow C$: on input of public parameters par, a receiver's public key pk and a plaintext m, this probabilistic algorithm outputs a first level ciphertext that cannot be re-encrypted for another party.

- $\mathsf{Enc}_2(\mathsf{par}, pk, m) \rightarrow C$: given public parameters par, a receiver's public key pk and a plaintext m, this randomized algorithm outputs a second level ciphertext that can be re-encrypted into a first level ciphertext (intended to a possibly different receiver) using the appropriate re-encryption key.

- $\mathsf{ReEnc}(\mathsf{par}, R_{ij}, C) \rightarrow C'$: this (possibly randomized) algorithm takes as input public parameters par, a re-encryption key R_{ij} and a second level ciphertext C encrypted under user i's public key. The output is a first level ciphertext C' re-encrypted for user j. In a single hop scheme, C' cannot be re-encrypted any further. If the well-formedness of C is publicly verifiable, the algorithm should output '`invalid`' whenever C is ill-formed w.r.t. X_i.

- $\mathsf{Dec}_1(\mathsf{par}, sk, C) \rightarrow m$: on input of a private key sk, a first level ciphertext C and system-wide parameters par, this algorithm outputs a message $m \in \{0,1\}^*$ or a distinguished message '`invalid`'.

- $\mathsf{Dec}_2(\mathsf{par}, sk, C) \rightarrow m$: given a private key sk, a second level ciphertext C and common public parameters par, this algorithm returns either a plaintext $m \in \{0,1\}^*$ or '`invalid`'.

Moreover, for any common public parameters par, for any message $m \in \{0,1\}^*$ and any couple of private/public key pair (sk_i, pk_i), (sk_j, pk_j) these algorithms should satisfy the following conditions of correctness:

$$\mathsf{Dec}_1(\mathsf{par}, sk_i, \mathsf{Enc}_1(\mathsf{par}, pk_i, m)) = m; \quad \mathsf{Dec}_2(\mathsf{par}, sk_i, \mathsf{Enc}_2(\mathsf{par}, pk_i, m)) = m;$$
$$\mathsf{Dec}_1(\mathsf{par}, sk_j, \mathsf{ReEnc}(\mathsf{par}, \mathsf{ReKeygen}(\mathsf{par}, sk_i, pk_j), \mathsf{Enc}_2(\mathsf{par}, pk_i, m))) = m.$$

To lighten notations, we will sometimes omit to explicitly write the set of common public parameters par, taken as input by all but one of the above algorithms.

Chosen-ciphertext security. The definition of chosen-ciphertext security that we consider is naturally inspired from the bidirectional case [CH07] which in turn extends ideas from Canetti, Krawczyk and Nielsen [CKN03] to the proxy re-encryption setting. For traditional public key cryptosystems, in this relaxation of Rackoff and Simon's definition [RS91], an adversary who can simply turn a given ciphertext into another encryption of the same plaintext is *not* deemed successful. In the game-based security definition, the attacker is notably disallowed to ask for a decryption of a re-randomized version of the challenge ciphertext. This relaxed notion was argued in [CKN03] to suffice for most practical applications.

Our definition considers a challenger that produces a number of public keys. As in [CH07], we do not allow the adversary to adaptively determine which parties will be compromised. On the other hand, we also allow her to adaptively query a re-encryption oracle and decryption oracles. A difference with [CH07] is that the adversary is directly provided with re-encryption keys that she is entitled to know (instead of leaving her adaptively request them as she likes). We also depart from [CH07], and rather follow [AFGH06], in that we let the target public key be determined by the challenger at the beginning of the game. Unlike [AFGH06], we allow the challenger to reveal re-encryption keys R_{ij} when j is corrupt for honest users i that differ from the target receiver. We insist that such an enhancement only makes sense for *single-hop* schemes like ours (as the adversary would trivially win the game if the scheme were multi-hop).

Definition A.2.2 A (single-hop) unidirectional PRE scheme is replayable chosen-ciphertext secure (RCCA) at level 2 if the probability

$$\begin{aligned}\Pr[(pk^\star, sk^\star) \leftarrow \mathsf{Keygen}(\lambda), \{(pk_x, sk_x) \leftarrow \mathsf{Keygen}(\lambda)\}, \{(pk_h, sk_h) \leftarrow \mathsf{Keygen}(\lambda)\},\\ \{R_{x\star} \leftarrow \mathsf{ReKeygen}(sk_x, pk^\star)\},\\ \{R_{\star h} \leftarrow \mathsf{ReKeygen}(sk^\star, pk_h)\}, \{R_{h\star} \leftarrow \mathsf{ReKeygen}(sk_h, pk^\star)\},\\ \{R_{hx} \leftarrow \mathsf{ReKeygen}(sk_h, pk_x)\}, \{R_{xh} \leftarrow \mathsf{ReKeygen}(sk_x, pk_h)\},\\ \{R_{hh'} \leftarrow \mathsf{ReKeygen}(sk_h, pk_{h'})\}, \{R_{xx'} \leftarrow \mathsf{ReKeygen}(sk_x, pk_{x'})\},\\ (m_0, m_1, St) \leftarrow \mathcal{A}^{\mathcal{O}_{1\text{-}dec}, \mathcal{O}_{renc}}(pk^\star, \{(pk_x, sk_x)\}, \{pk_h\}, \{R_{x\star}\}, \{R_{h\star}\},\\ \{R_{\star h}\}, \{R_{xh}\}, \{R_{hx}\}, \{R_{hh'}\}, \{R_{xx'}\}),\\ d^\star \overset{R}{\leftarrow} \{0,1\}, C^\star = \mathsf{Enc}_2(m_{d^\star}, pk^\star), d' \leftarrow \mathcal{A}^{\mathcal{O}_{1\text{-}dec}, \mathcal{O}_{renc}}(C^\star, St) :\\ d' = d^\star]\end{aligned}$$

is negligibly (as a function of the security parameter λ) close to 1/2 for any PPT adversary $\mathcal{A}$. In our notation, St is a state information maintained by $\mathcal{A}$ while $(pk^\star, sk^\star)$ is the target user's key pair generated by the challenger that also chooses other keys for corrupt and honest parties. For other honest parties, keys are subscripted by h or h' and we subscript corrupt keys by x or x'. The adversary is given access to all re-encryption keys but those that would allow re-encrypting from the target user to a corrupt one. In the game, $\mathcal{A}$ is said to have advantage ε if this probability, taken over random choices of $\mathcal{A}$ and all oracles, is at least $1/2 + \varepsilon$. Oracles $\mathcal{O}_{1\text{-}dec}, \mathcal{O}_{renc}$ proceed as follows:

Re-encryption $\mathcal{O}_{renc}$: on input (pk_i, pk_j, C), where C is a second level ciphertext and pk_i, pk_j were produced by Keygen, this oracle responds with '`invalid`' if C is not properly shaped w.r.t. pk_i. It returns a special symbol $\perp$ if pk_j is corrupt and $(pk_i, C) = (pk^\star, C^\star)$. Otherwise, the re-encrypted first level ciphertext $C' = \mathsf{ReEnc}(\mathsf{ReKeygen}(sk_i, pk_j), C)$ is returned to $\mathcal{A}$.

First level decryption oracle $\mathcal{O}_{1\text{-}dec}$: given a pair (pk, C), where C is a first level ciphertext and pk was produced by Keygen, this oracle returns '`invalid`' if C is ill-formed w.r.t. pk. If the query occurs in the post-challenge phase (a.k.a. "guess" stage as opposed to the "find" stage), it outputs a special symbol $\perp$ if (pk, C) is a $\mathsf{Derivative}$ of the challenge pair $(pk^\star, C^\star)$. Otherwise, the plaintext $m = \mathsf{Dec}_1(sk, C)$ is revealed to $\mathcal{A}$. $\mathsf{Derivative}$*s* of $(pk^\star, C^\star)$ are defined as follows.

> If C is a first level ciphertext and $pk = pk^\star$ or pk is another honest user, (pk, C) is a $\mathsf{Derivative}$ of $(pk^\star, C^\star)$ if $\mathsf{Dec}_1(sk, C) \in \{m_0, m_1\}$.

Explicitly providing the adversary with a second level decryption oracle is useless. Indeed, ciphertexts encrypted under public keys from $\{pk_h\}$ can be re-encrypted for corrupt users given the set $\{R_{hx}\}$. Besides, second level encryptions under $pk^\star$ can be translated for other honest users using $\{R_{\star h}\}$. The resulting first level ciphertext can then be queried for decryption at the first level.

Security of first level ciphertexts. The above definition provides adversaries with a second level ciphertext in the challenge phase. An orthogonal definition of security captures their inability to distinguish first level ciphertexts as well. For *single-hop* schemes, the adversary is

granted access to *all* re-encryption keys in this definition. Since first level ciphertexts cannot be re-encrypted, there is indeed no reason to keep attackers from obtaining all honest-to-corrupt re-encryption keys. The re-encryption oracle thus becomes useless since all re-encryption keys are available to $\mathcal{A}$. For the same reason, a second level decryption oracle is also unnecessary. Finally, Derivatives of the challenge ciphertext are simply defined as encryptions of either m_0 or m_1 for the same target public key $pk^\star$. A unidirectional PRE scheme is said RCCA-secure at level 1 if it satisfies this notion.

Remark 1. As in [CH07], we assume a static corruption model. Proving security against adaptive corruptions turns out to be more challenging. In our model and the one of [CH07], the challenger generates public keys for all parties and allows the adversary to obtain private keys for some of them. This does not capture a scenario where adversaries generate public keys on behalf of corrupt parties (possibly non-uniformly or as a function of honest parties' public keys) themselves. We also leave open the problem of achieving security in such a setting.

Remark 2. A possible enhancement of definition A.2.2 is to allow adversaries to adaptively choose the target user at the challenge phase within the set of honest players. After having selected a set of corrupt parties among n players at the beginning, the adversary receives a set of n public keys, private keys of corrupt users as well as corrupt-to-corrupt, corrupt-to-honest and honest-to-honest re-encryption keys. When she outputs messages (m_0, m_1) and the index $i^\star$ of a honest user in the challenge step, she obtains an encryption of $m_{d^\star}$ under $pk_{i^\star}$ together with all honest-to-corrupt re-encryption keys R_{ij} with $i \neq i^\star$.

In this setting, a second level decryption oracle is also superfluous for schemes (like ours) where second level ciphertexts can be publicly turned into first level encryptions of the same plaintext for the same receiver. The scheme that we describe remains secure in this model at the expense of a probability of failure for the simulator that has to foresee which honest user will be attacked with probability $O(1/n)$.

Master secret security. In [AFGH06], Ateniese *et al.* define another important security requirement for unidirectional PRE schemes. This notion, termed *master secret security*, demands that no coalition of dishonest delegatees be able to pool their re-encryption keys in order to expose the private key of their common delegator. More formally, the following probability should be negligible as a function of the security parameter λ.

$$\begin{aligned}\Pr[(pk^\star, sk^\star) \leftarrow \mathsf{Keygen}(\lambda), \quad & \{(pk_x, sk_x) \leftarrow \mathsf{Keygen}(\lambda)\},\\ & \{R_{\star x} \leftarrow \mathsf{ReKeygen}(sk^\star, pk_x)\},\\ & \{R_{x\star} \leftarrow \mathsf{ReKeygen}(sk_x, pk^\star)\},\\ & \gamma \leftarrow A(pk^\star, \{(pk_x, sk_x)\}, \{R_{\star x}\}, \{R_{x\star}\})\\ & : \gamma = sk^\star]\end{aligned}$$

At first glance, this notion might seem too weak in that it does not consider colluding delegatees who would rather undertake to produce a new re-encryption key $R_{\star x'}$ that was not originally given and allows re-encrypting from the target user to another malicious party x'. As stressed in [AFGH06] however, *all* known unidirectional PRE schemes fail to satisfy such a stronger notion of security. It indeed remains an open problem to construct a scheme withstanding this kind of *transfer of delegation* attack.

The notion of RCCA security at the first level is easily seen to imply the master secret security and we will only discuss the former.

A.2.2 Bilinear Maps and Complexity Assumptions

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. efficient computability for any input pair;
3. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

We shall assume the intractability of a variant of the Decision Bilinear Diffie-Hellman problem.

Definition A.2.3 The **3-Quotient Decision Bilinear Diffie-Hellman** (3-QDBDH) assumption posits the hardness of distinguishing $e(g, g)^{b/a}$ from random given $(g, g^a, g^{(a^2)}, g^{(a^3)}, g^b)$. A distinguisher $\mathcal{B}$ (t, ε)-breaks the assumption if it runs in time t and

$$\begin{aligned}\big|\Pr[\mathcal{B}(g, g^a, g^{(a^2)}, g^{(a^3)}, g^b, e(g, g)^{b/a}) = 1 | a, b \xleftarrow{R} \mathbb{Z}_p^*] \\ - \Pr[\mathcal{B}(g, g^a, g^{(a^2)}, g^{(a^3)}, g^b, e(g, g)^z) = 1 | a, b, z \xleftarrow{R} \mathbb{Z}_p^*]\big| \geq \varepsilon.\end{aligned}$$

The 3-QDBDH problem is obviously not easier than the (q-DBDHI) problem [BB04a] for $q \geq 3$, which is to recognize $e(g, g)^{1/a}$ given $(g, g^a, \ldots, g^{(a^q)}) \in \mathbb{G}^{q+1}$. Dodis and Yampolskiy showed that this problem was indeed hard in generic groups [DY05]. Their result thus implies the hardness of 3-QDBDH in generic groups.

Moreover, its intractability for any polynomial time algorithm can be classified among *mild* decisional assumptions (according to [BW06a]) as its strength does not depend on the number of queries allowed to adversaries whatsoever.

A.2.3 One-time signatures

As an underlying tool for applying the Canetti-Halevi-Katz methodology [CHK04, BCHK07], we need one-time signatures. Such a primitive consists of a triple of algorithms $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ such that, on input of a security parameter λ, $\mathcal{G}$ generates a one-time key pair (ssk, svk) while, for any message M, $\mathcal{V}(\sigma, svk, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0 otherwise.

As in [CHK04], we need strongly unforgeable one-time signatures, which means that no PPT adversary can create a new signature for a previously signed message (according to [ADR02]).

Definition A.2.4 $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strong one-time signature if the probability

$$\begin{aligned}Adv^{\mathsf{OTS}} = \Pr\big[\quad & (ssk, svk) \leftarrow \mathcal{G}(\lambda); (M, St) \leftarrow \mathcal{F}(svk); \\ & \sigma \leftarrow \mathcal{S}(ssk, M); (M', \sigma') \leftarrow \mathcal{F}(M, \sigma, svk, St) : \\ & \mathcal{V}(\sigma', svk, M') = 1 \wedge (M', \sigma') \neq (M, \sigma) \ \big],\end{aligned}$$

where St denotes the state information maintained by $\mathcal{F}$ between stages, is negligible for any PPT forger $\mathcal{F}$.

A.3 The Scheme

Our construction is inspired from the first unidirectional scheme suggested in [AFGH06] where second level ciphertexts $(C_1, C_2) = (X^r, m \cdot e(g,g)^r)$, that are encrypted under the public key $X = g^x$, can be re-encrypted into first level ciphertexts $(e(C_1, R_{xy}), C_2) = (e(g,g)^{ry}, m \cdot e(g,g)^r)$ using the re-encryption key $R_{xy} = g^{y/x}$. Using his private key y s.t. $Y = g^y$, the receiver can then obtain the message.

The Canetti-Hohenberger method for achieving CCA-security for proxy re-encryption borrows from [CHK04, BMW05, Kil06] in that it appends to the ciphertext a checksum value consisting of an element of $\mathbb{G}$ raised to the random encryption exponent r. In the security proof, the simulator uses the publicly verifiable validity of ciphertexts in groups equipped with bilinear maps. Unfortunately, the same technique does not directly apply to secure the unidirectional PRE scheme of [AFGH06] against chosen-ciphertext attacks. The difficulty is that, after re-encryption, level 1 ciphertexts have one component in the target group $\mathbb{G}_T$ and pairings cannot be used any longer to check the equality of two discrete logarithms in groups $\mathbb{G}$ and $\mathbb{G}_T$. Therefore, the simulator cannot tell apart well-shaped level 1 ciphertexts from invalid ones.

The above technical issue is addressed by having the proxy replace C_1 with a pair $(C_1', C_1'') = (R_{xy}^{1/t}, C_1^t) = (g^{y/(tx)}, X^{rt})$, for a randomly chosen "blinding exponent" $t \xleftarrow{R} \mathbb{Z}_p^*$ that hides the re-encryption key in C_1', in such a way that all ciphertext components but C_2 remain in $\mathbb{G}$. This still allows the second receiver holding y s.t. $Y = g^y$ to compute $m = C_2 / e(C_1', C_1'')^{1/y}$. To retain the publicly verifiable well-formedness of re-encrypted ciphertexts however, the proxy needs to include X^t in the ciphertext so as to prove the consistency of the encryption exponent r w.r.t. the checksum value.

Of course, since the re-encryption algorithm is probabilistic, many first level ciphertexts may correspond to the same second level one. For this reason, we need to tolerate a harmless form of malleability (akin to those accepted as reasonable in [ADR02, CKN03, Sho01]) of ciphertexts at level 1.

A.3.1 Description

Our system is reminiscent of the public key cryptosystem obtained by applying the Canetti-Halevi-Katz transform [CHK04] to the second selective-ID secure identity-based encryption scheme described in [BB04a][1].

Like the Canetti-Hohenberger construction [CH07], the present scheme uses a strongly unforgeable one-time signature to tie several ciphertext components altogether and offer a safeguard against chosen-ciphertext attacks in the fashion of Canetti, Halevi and Katz [CHK04]. For simplicity, the description below assumes that verification keys of the one-time signature are encoded as elements from $\mathbb{Z}_p^*$. In practice, such verification keys are typically much longer than $|p|$ and a collision-resistant hash function should be applied to map them onto $\mathbb{Z}_p^*$.

Global-setup(λ)**:** given a security parameter λ, choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, generators $g, u, v \xleftarrow{R} \mathbb{G}$ and a strongly unforgeable one-time signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global parameters are

$$\mathsf{par} := \{\mathbb{G}, \mathbb{G}_T, g, u, v, \mathsf{Sig}\}.$$

Keygen(λ)**:** user i sets his public key as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

[1] It was actually shown in [Kil06] that, although the security of the underlying IBE scheme relies on a rather strong assumption, a weaker assumption such as the one considered here was sufficient to prove the security of the resulting public key encryption scheme.

ReKeygen(x_i, X_j)**:** given user i's private key x_i and user j's public key X_j, generate the unidirectional re-encryption key $R_{ij} = X_j^{1/x_i} = g^{x_j/x_i}$.

Enc$_1(m, X_i, \mathsf{par})$**:** to encrypt a message $m \in \mathbb{G}_T$ under the public key X_i at the first level, the sender proceeds as follows.

1. Select a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Pick $r, t \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2' = X_i^t \quad C_2'' = g^{1/t} \quad C_2''' = X_i^{rt} \quad C_3 = e(g,g)^r \cdot m \qquad C_4 = (u^{svk} \cdot v)^r$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (C_3, C_4))$ on (C_3, C_4).

The ciphertext is $C_i = (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$.

Enc$_2(m, X_i, \mathsf{par})$**:** to encrypt a message $m \in \mathbb{G}_T$ under the public key X_i at level 2, the sender conducts the following steps.

1. Select a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Choose $r \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2 = X_i^r \qquad C_3 = e(g,g)^r \cdot m \qquad C_4 = (u^{svk} \cdot v)^r$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (C_3, C_4))$ on the pair (C_3, C_4).

The ciphertext is $C_i = (C_1, C_2, C_3, C_4, \sigma)$.

ReEnc(R_{ij}, C_i)**:** on input of the re-encryption key $R_{ij} = g^{x_j/x_i}$ and a ciphertext

$$C_i = (C_1, C_2, C_3, C_4, \sigma),$$

check the validity of the latter by testing the following conditions

$$e(C_2, u^{C_1} \cdot v) = e(X_i, C_4) \tag{A.1}$$

$$\mathcal{V}(C_1, \sigma, (C_3, C_4)) = 1. \tag{A.2}$$

If well-formed, C_i is re-encrypted by choosing $t \xleftarrow{R} \mathbb{Z}_p^*$ and computing

$$C_2' = X_i^t \qquad C_2'' = R_{ij}^{1/t} = g^{(x_j/x_i)t^{-1}} \qquad C_2''' = C_2^t = X_i^{rt}$$

The re-encrypted ciphertext is

$$C_j \;=\; (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma).$$

If ill-formed, C_i is declared '`invalid`'.

$\mathsf{Dec}_1(C_j, sk_j)$: the validity of a level 1 ciphertext C_j is checked by testing if

$$e(C'_2, C''_2) = e(X_j, g) \tag{A.3}$$

$$e(C'''_2, u^{C_1} \cdot v) = e(C'_2, C_4) \tag{A.4}$$

$$\mathcal{V}(C_1, \sigma, (C_3, C_4)) = 1 \tag{A.5}$$

If relations (A.3)-(A.5) hold, the plaintext $m = C_3/e(C''_2, C'''_2)^{1/x_j}$ is returned. Otherwise, the algorithm outputs '`invalid`'.

$\mathsf{Dec}_2(C_i, sk_i)$: if the level 2 ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$ satisfies relations (A.1)-(A.2), receiver i can obtain $m = C_3/e(C_2, g)^{1/x_i}$. The algorithm outputs '`invalid`' otherwise.

Outputs of the re-encryption algorithm are perfectly indistinguishable from level 1 ciphertexts produced by the sender. Indeed, if $\tilde{t} = tx_i/x_j$, we can write

$$C'_2 = X_i^t = X_j^{\tilde{t}} \qquad C''_2 = g^{(x_j/x_i)t^{-1}} = g^{\tilde{t}^{-1}} \qquad C'''_3 = X_i^{rt} = X_j^{r\tilde{t}}.$$

As in the original scheme described in [AFGH06], second level ciphertexts can be publicly turned into first level ciphertexts encrypted for the same receiver if the identity element of $\mathbb{G}$ is used as a re-encryption key.

In the first level decryption algorithm, relations (A.3)-(A.5) guarantee that re-encrypted ciphertexts have the correct shape. Indeed, since $C_4 = (u^{C_1} \cdot v)^r$ for some unknown exponent $r \in \mathbb{Z}_p$, equality (A.4) implies that $C'''_2 = C''^r_2$. From (A.3), it comes that $e(C''_2, C'''_2) = e(X_j, g)^r$.

We finally note that first level ciphertexts can be publicly re-randomized by changing the triple (C'_2, C''_2, C''_3) into $(C'^s_2, C''^{1/s}_2, C'''^s_3)$ for a random $s \in \mathbb{Z}_p^*$. However, the pairing value $e(C''_2, C'''_2)$ remains constant and, re-randomizations of a given first level ciphertext are publicly detectable.

A.3.2 Security

For convenience, we will prove security under an equivalent formulation of the 3-QDBDH assumption.

Lemma A.3.1 The 3-QDBDH problem is equivalent to decide whether T equals $e(g, g)^{b/a^2}$ or a random value given $(g, g^{1/a}, g^a, g^{(a^2)}, g^b)$ as input.

Proof: Given $(g, g^{1/a}, g^a, g^{(a^2)}, g^b)$, we can build a 3-QDBDH instance by setting $(y = g^{1/a}, y^A = g, y^{(A^2)} = g^a, y^{(A^3)} = g^{(a^2)}, y^B = g^b)$, which implicitly defines $A = a$ and $B = ab$. Then, we have $e(y, y)^{B/A} = e(g^{1/a}, g^{1/a})^{(ab)/a} = e(g, g)^{b/a^2}$. The converse implication is easily established and demonstrates the equivalence between both problems. ∎

Theorem A.3.2 Assuming the strong unforgeability of the one-time signature, the scheme is RCCA-secure at level 2 under the 3-QDBDH assumption.

Proof: Let $(A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{(a^2)}, B = g^b, T)$ be a modified 3-QDBDH instance. We construct an algorithm $\mathcal{B}$ deciding whether $T = e(g, g)^{b/a^2}$ out of a successful RCCA adversary $\mathcal{A}$.

Before describing $\mathcal{B}$, we first define an event F_{OTS} and bound its probability to occur. Let $C^\star = (svk^\star, C_2^\star, C_3^\star, C_4^\star, \sigma^\star)$ denote the challenge ciphertext given to $\mathcal{A}$ in the game.

Let F_{OTS} denote the event that, in the security game, $\mathcal{A}$ issues a decryption query for a first level ciphertext $C = (svk^\star, C_2', C_2'', C_2''', C_3, C_4, \sigma)$ or a re-encryption query $C = (svk^\star, C_2, C_3, C_4, \sigma)$ where $(C_3, C_4, \sigma) \neq (C_3^\star, C_4^\star, \sigma^\star)$ but $\mathcal{V}(\sigma, svk, (C_3, C_4)) = 1$. In the "find" stage, $\mathcal{A}$ has simply no information on $svk^\star$. Hence, the probability of a pre-challenge occurrence of F_{OTS} does not exceed $q_O \cdot \delta$ if q_O is the overall number of oracle queries and δ denotes the maximal probability (which by assumption does not exceed $1/p$) that any one-time verification key svk is output by $\mathcal{G}$. In the "guess" stage, F_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore, the probability $\Pr[F_{\mathsf{OTS}}] \leq q_O/p + Adv^{\mathsf{OTS}}$, where the second term accounts for the probability of definition A.2.4, must be negligible by assumption. We now proceed with the description of $\mathcal{B}$ that simply halts and outputs a random bit if F_{OTS} occurs. In a preparation phase, $\mathcal{B}$ generates a one-time signature key pair $(ssk^\star, svk^\star) \leftarrow \mathcal{G}(\lambda)$ and provides $\mathcal{A}$ with public parameters including $u = A_1^{\alpha_1}$ and $v = A_1^{-\alpha_1 svk^\star} \cdot A_2^{\alpha_2}$ for random $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_p^*$. Observe that u and v define a "hash function" $F(svk) = u^{svk} \cdot v = A_1^{\alpha_1(svk - svk^\star)} \cdot A_2^{\alpha_2}$. In the following, we call HU the set of honest parties, including user $i^\star$ that is assigned the target public key $pk^\star$, and CU the set of corrupt parties. Throughout the game, $\mathcal{A}$'s environment is simulated as follows.

- *Key generation*: public keys of honest users $i \in HU \backslash \{i^\star\}$ are defined as $X_i = A_1^{x_i} = g^{a x_i}$ for a randomly chosen $x_i \xleftarrow{R} \mathbb{Z}_p^*$. The target user's public key is set as $X_{i^\star} = A_2^{x_{i^\star}} = g^{(x_{i^\star} a^2)}$ with $x_{i^\star} \xleftarrow{R} \mathbb{Z}_p^*$. The key pair of a corrupt user $i \in CU$ is set as $(X_i = g^{x_i}, x_i)$, for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$, so that (X_i, x_i) can be given to $\mathcal{A}$. To generate re-encryption keys R_{ij} from player i to player j, $\mathcal{B}$ has to distinguish several situations:

  - If $i \in CU$, $\mathcal{B}$ knows $sk_i = x_i$. Given X_j, it simply outputs X_j^{1/x_i}.
  - If $i \in HU \backslash \{i^\star\}$ and $j = i^\star$, $\mathcal{B}$ returns $R_{ii^\star} = A_1^{x_{i^\star}/x_i} = g^{x_{i^\star} a^2/(a x_i)}$ which is a valid re-encryption key.
  - If $i = i^\star$ and $j \in HU \backslash \{i^\star\}$, $\mathcal{B}$ responds with $R_{i^\star j} = A_{-1}^{x_i/x_{i^\star}} = g^{(a x_i/(x_{i^\star} a^2))}$ that has also the correct distribution.
  - If $i, j \in HU \backslash \{i^\star\}$, $\mathcal{B}$ returns $R_{ij} = g^{x_j/x_i} = g^{(a x_j)/(a x_i)}$.
  - If $i \in HU \backslash \{i^\star\}$ and $j \in CU$, $\mathcal{B}$ outputs $R_{ij} = A_{-1}^{x_j/x_i} = g^{x_j/(a x_i)}$ which is also computable.

- *Re-encryption* queries: when facing a re-encryption query from user i to user j for a second level ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$, $\mathcal{B}$ returns '`invalid`' if relations (A.1)-(A.2) are not satisfied.

  - If $i \neq i^\star$ or if $i = i^\star$ and $j \in HU \backslash \{i^\star\}$, $\mathcal{B}$ simply re-encrypts using the re-encryption key R_{ij} which is available in either case.
  - If $i = i^\star$ and $j \in CU$,
    - If $C_1 = svk^\star$, $\mathcal{B}$ is faced with an occurrence of F_{OTS} and halts. Indeed, re-encryptions of the challenge ciphertext towards corrupt users are disallowed in the "guess" stage. Therefore, $(C_3, C_4, \sigma) \neq (C_3^\star, C_4^\star, \sigma^\star)$ since we would have $C_2 \neq C_2^\star$ and $i \neq i^\star$ if $(C_3, C_4, \sigma) = (C_3^\star, C_4^\star, \sigma^\star)$.

· We are thus left with the case $C_1 \neq svk^\star$, $i = i^\star$ and $j \in CU$. Given $C_2^{1/x_{i^\star}} = A_2^r$, from $C_4 = F(svk)^r = (A_1^{\alpha_1(svk-svk^\star)} \cdot A_2^{\alpha_2})^r$, $\mathcal{B}$ can compute

$$A_1^r = (g^a)^r = \left(\frac{C_4}{C_2^{\alpha_2/x_{i^\star}}}\right)^{\frac{1}{\alpha_1(svk-svk^\star)}}. \tag{A.6}$$

Knowing g^{ar} and user j's private key x_j, $\mathcal{B}$ picks $t \xleftarrow{R} \mathbb{Z}_p^*$ to compute

$$C_2' = A_1^t = g^{at} \qquad C_2'' = A_{-1}^{x_j/t} = (g^{1/a})^{x_j/t} \qquad C_2''' = (A_1^r)^t = (g^{ar})^t$$

and return $C_j = (C_1, C_2', C_2'', C_3''', C_3, C_4, \sigma)$ which has the proper distribution. Indeed, if we set $\tilde{t} = at/x_j$, we have $C_2' = X_j^{\tilde{t}}$, $C_2'' = g^{1/\tilde{t}}$ and $C_2''' = X_j^{r\tilde{t}}$.

- *First level decryption* queries: when the decryption of a first level ciphertext

$$C_j = (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$$

is queried under a public key X_j, $\mathcal{B}$ returns '`invalid`' if relations (A.3)-(A.5) do not hold. We assume that $j \in HU$ since $\mathcal{B}$ can decrypt using the known private key otherwise. Let us first assume that $C_1 = C_1^\star = svk^\star$. If $(C_3, C_4, \sigma) \neq (C_3^\star, C_4^\star, \sigma^\star)$, $\mathcal{B}$ is presented with an occurrence of F_{OTS} and halts. If $(C_3, C_4, \sigma) = (C_3^\star, C_4^\star, \sigma^\star)$, $\mathcal{B}$ outputs $\perp$ which deems C_j as a Derivative of the challenge pair $(C^\star, X_{i^\star})$. Indeed, it must be the case that $e(C_2'', C_2''') = e(g, X_j)^r$ for the same underlying exponent r as in the challenge phase. We now assume $C_1 \neq svk^\star$.

- If $j \in HU \backslash \{i^\star\}$, $X_j = g^{ax_j}$ for a known $x_j \in \mathbb{Z}_p^*$. The validity of the ciphertext ensures that $e(C_2'', C_2''') = e(X_j, g)^r = e(g, g)^{arx_j}$ and $C_4 = F(svk)^r = g^{\alpha_1 ar(svk-svk^\star)} \cdot g^{a^2 r\alpha_2}$ for some $r \in \mathbb{Z}_p$. Therefore,

$$e(C_4, A_{-1}) = e(C_4, g^{1/a}) = e(g, g)^{\alpha_1 r(svk-svk^\star)} \cdot e(g, g)^{ar\alpha_2} \tag{A.7}$$

and

$$e(g, g)^r = \left(\frac{e(C_4, A_{-1})}{e(C_2'', C_2''')^{\alpha_2/x_j}}\right)^{\frac{1}{\alpha_1(svk-svk^\star)}} \tag{A.8}$$

reveals the plaintext m since $svk \neq svk^\star$.

- If $j = i^\star$, we have $X_j = g^{(x_{i^\star}a^2)}$ for a known exponent $x_{i^\star} \in \mathbb{Z}_p^*$. Since $e(C_2'', C_2''') = e(X_{i^\star}, g)^r = e(g, g)^{a^2 rx_{i^\star}}$ and

$$e(C_4, g) = e(g, g)^{\alpha_1 ar(svk-svk^\star)} \cdot e(g, g)^{a^2 r\alpha_2},$$

$\mathcal{B}$ can first obtain

$$\gamma = e(g, g)^{ar} = \left(\frac{e(C_4, g)}{e(C_2'', C_2''')^{\alpha_2/x_{i^\star}}}\right)^{\frac{1}{\alpha_1(svk-svk^\star)}}.$$

Together with relation (A.7), γ in turn uncovers

$$e(g, g)^r = \left(\frac{e(C_4, A_{-1})}{\gamma^{\alpha_2/x_{i^\star}}}\right)^{\frac{1}{\alpha_1(svk-svk^\star)}}$$

and the plaintext $m = C_3/e(g, g)^r$.

In the "guess" stage, $\mathcal{B}$ must check that m differs from messages m_0, m_1 involved in the challenge query. If $m \in \{m_0, m_1\}$, $\mathcal{B}$ returns $\perp$ according to the replayable CCA-security rules.

- *Challenge*: when she decides that the first phase is over, $\mathcal{A}$ chooses messages (m_0, m_1). At this stage, $\mathcal{B}$ flips a coin $d^\star \xleftarrow{R} \{0,1\}$ and sets the challenge ciphertext as

$$C_1^\star = svk^\star \quad C_2^\star = B^{x_{i^\star}} \quad C_3^\star = m_{d^\star} \cdot T \quad C_4^\star = B^{\alpha_2}$$

and $\sigma = \mathcal{S}(ssk^\star, (C_3, C_4))$.

Since $X_{i^\star} = A_2^{x_{i^\star}} = g^{x_{i^\star} a^2}$ and $B = g^b$, $C^\star$ is a valid encryption of $m_{d^\star}$ with the random exponent $r = b/a^2$ if $T = e(g,g)^{b/a^2}$. In contrast, if T is random in $\mathbb{G}_T$, $C^\star$ perfectly hides $m_{d^\star}$ and $\mathcal{A}$ cannot guess $d^\star$ with better probability than $1/2$. When $\mathcal{A}$ eventually outputs her result $d' \in \{0,1\}$, $\mathcal{B}$ decides that $T = e(g,g)^{b/a^2}$ if $d' = d^\star$ and that T is random otherwise. ▮

Theorem A.3.3 Assuming the strong unforgeability of the one-time signature, the scheme is RCCA-secure at level 1 under the 3-QDBDH assumption.

Proof: The proof is very similar to the one of theorem A.3.2. Given a 3-QDBDH instance $(A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{(a^2)}, B = g^b, T)$, we construct an algorithm $\mathcal{B}$ that decides if $T = e(g,g)^{b/a^2}$.
Before describing $\mathcal{B}$, we consider the same event F_{OTS} as in the proof of theorem A.3.2 except that it can only arise during a decryption query (since there is no re-encryption oracle). Assuming the strong unforgeability of the one-time signature, such an event occurs with negligible probability as detailed in the proof of theorem A.3.2. We can now describe our simulator $\mathcal{B}$ that simply halts and outputs a random bit if F_{OTS} ever occurs. Let also $C^\star = (C_1^\star, C_2'^\star, C_2''^\star, C_2'''^\star, C_3^\star, C_4^\star, \sigma^\star)$ denote the challenge ciphertext at the first level.
Algorithm $\mathcal{B}$ generates a one-time signature key pair $(ssk^\star, svk^\star) \leftarrow \mathcal{G}(\lambda)$ and the same public parameters as in theorem A.3.2. Namely, it sets $u = A_1^{\alpha_1}$ and $v = A_1^{-\alpha_1 svk^\star} \cdot A_2^{\alpha_2}$ with $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_p^*$ so that $F(svk) = u^{svk} \cdot v = A_1^{\alpha_1(svk - svk^\star)} \cdot A_2^{\alpha_2}$. As in the proof of theorem A.3.2, $i^\star$ identifies the target receiver. The attack environment is simulated as follows.

- *Key generation*: for corrupt users $i \in CU$ and almost all honest ones $i \in HU \backslash \{i^\star\}$, $\mathcal{B}$ sets $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$. The target user's public key is defined as $X_{i^\star} = A_1$. For corrupt users $i \in CU$, X_i and x_i are both revealed. All re-encryption keys are computable and given to $\mathcal{A}$. Namely, $R_{ij} = g^{x_j/x_i}$ if $i, j \neq i^\star$; $R_{i^\star j} = A_{-1}^{x_j}$ and $R_{ji^\star} = A_1^{1/x_j}$ for $j \neq i^\star$.

- *First level decryption* queries: when the decryption of a ciphertext

$$C_j = (C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$$

is queried for a public key X_j, $\mathcal{B}$ returns '`invalid`' if relations (A.3)-(A.5) do not hold. We assume that $j = i^\star$ since $\mathcal{B}$ can decrypt using the known private key x_j otherwise. We have $C_2' = A_1^t$, $C_2'' = g^{1/t}$, $C_2''' = A_1^{rt}$ for unknown exponents $r, t \in \mathbb{Z}_p^*$. Since $e(C_2'', C_2''') = e(g,g)^{ar}$ and

$$e(C_4, A_{-1}) = e(g,g)^{\alpha_1 r(svk - svk^\star)} \cdot e(g,g)^{ar\alpha_2},$$

$\mathcal{B}$ can obtain

$$e(g,g)^r = \left(\frac{e(C_4, A_{-1})}{e(C_2'', C_2''')^{\alpha_2}}\right)^{\frac{1}{\alpha_1(svk - svk^\star)}}$$

which reveals the plaintext $m = C_3/e(g,g)^r$ as long as $svk \neq svk^\star$. In the event that $C_1 = svk^\star$ in a post-challenge query,

- If $e(C_2'', C_2''') = e(C_2''^\star, C_2'''^\star)$ then $\mathcal{B}$ returns $\perp$, meaning that C_j is simply a re-randomization (and thus a Derivative) of the challenge ciphertext.
- Otherwise, we necessarily have $(C_3^\star, C_4^\star, \sigma^\star) \neq (C_3, C_4, \sigma)$, which is an occurrence of F_{OTS} and implies $\mathcal{B}$'s termination.

In the "guess" stage, $\mathcal{B}$ must ensure that m differs from messages m_0, m_1 of the challenge phase before answering the query.

- *Challenge*: when the first phase is over, $\mathcal{A}$ outputs messages (m_0, m_1) and $\mathcal{B}$ flips a bit $d^\star \xleftarrow{R} \{0,1\}$. Then, it chooses $\mu \xleftarrow{R} \mathbb{Z}_p^*$ and sets

$$C_2'^\star = A_2^\mu \quad C_2''^\star = A_{-1}^{1/\mu} \quad C_2'''^\star = B^\mu$$
$$C_1^\star = svk^\star \quad C_3^\star = m_{d^\star} \cdot T \quad C_4^\star = B^{\alpha_2}$$

and $\sigma = \mathcal{S}(ssk^\star, (C_3, C_4))$.

Since $X_{i^\star} = A_1$ and $B = g^b$, $C^\star$ is a valid encryption of $m_{d^\star}$ with the random exponents $r = b/a^2$ and $t = a\mu$ whenever $T = e(g,g)^{b/a^2}$. When T is random, $C^\star$ perfectly hides $m_{d^\star}$ and $\mathcal{A}$ cannot guess $d^\star$ with better probability than $1/2$. Eventually, $\mathcal{B}$ bets that $T = e(g,g)^{b/a^2}$ if $\mathcal{A}$ correctly guesses $d^\star$ and that T is random otherwise. ▌

A.3.3 Efficiency

The first level decryption algorithm can be optimized using ideas from [Kil06, KG06]. Namely, verification tests (A.3)-(A.4) can be simultaneously achieved with high confidence by the receiver who can choose a random $\alpha \xleftarrow{R} \mathbb{Z}_p^*$ and test whether

$$\frac{e(C_2', C_2'' \cdot C_4^\alpha)}{e(C_2''', u^{svk} \cdot v)^\alpha} = e(g,g)^{x_j}.$$

Hence, computing a quotient of two pairings (which is faster than evaluating two independent pairings [GS06]) and two extra exponentiations suffice to check the validity of the ciphertext.

It could also be desirable to shorten ciphertexts that are significantly lengthened by one-time signatures and their public keys. To this end, ideas from Boneh and Katz [BK05] can be used as well as those of Boyen, Mei and Waters [BMW05]. In the latter case, ciphertexts can be made fairly compact as components C_1 and σ become unnecessary if the checksum value C_4 is computed using the Waters "hashing" technique [Wat05] applied to a collision-resistant hash of C_3. This improvement in the ciphertext size unfortunately comes at the expense of a long public key (made of about 160 elements of $\mathbb{G}$ as in [Wat05]) and a loose reduction.

A.4 A Scheme with Temporary Delegation

This section describes a variant of our scheme supporting temporary delegation. Like the temporary unidirectional PRE suggested in [AFGH06], it only allows the proxy to re-encrypt messages from A to B during a limited time period. If the scheme must be set up for T periods, we assume that a trusted server publishes randomly chosen elements $(h_1, \ldots, h_T) \in \mathbb{G}^T$ as global parameters. Alternatively, the server could publish a new value h_i that erases h_{i-1} at period i so as to keep short public parameters.

Global-setup(λ, T)**:** is as in section A.3 with the difference that additional random group elements $h_1, \ldots, h_T$ (where T is the number of time intervals that the scheme must be prepared for) are chosen. Global parameters are

$$\mathsf{par} := \{\mathbb{G}, \mathbb{G}_T, g, u, v, h_1, \ldots, h_T, \mathsf{Sig}\}.$$

Keygen(λ)**:** user i's public key is set as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

ReKeygen$(x_i, D_{(\ell,j)})$**:** when user j is willing to accept delegations during period $\ell \in \{1, \ldots, T\}$, he publishes a delegation acceptance value $D_{(\ell,j)} = h_\ell^{x_j}$. Given his private key x_i, user i then generates the temporary re-encryption key is $R_{ij\ell} = D_{(\ell,j)}^{1/x_i} = h_\ell^{x_j/x_i}$.

Enc$_1(m, X_i, \ell, \mathsf{par})$**:** to encrypt $m \in \mathbb{G}_T$ under the public key X_i at the first level during period $\ell \in \{1, \ldots, T\}$, the sender conducts the following steps.

1. Choose a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$; set $C_1 = svk$.
2. Pick $r, t \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2' = X_i^t \quad C_2'' = h_\ell^{1/t} \quad C_2''' = X_i^{rt} \quad C_3 = e(g, h_\ell)^r \cdot m \qquad C_4 = (u^{svk} \cdot v)^r$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (\ell, C_3, C_4))$ on (ℓ, C_3, C_4).

The ciphertext is $C_i = (\ell, C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$.

Enc$_2(m, X_i, \ell, \mathsf{par})$**:** to encrypt $m \in \mathbb{G}_T$ under the public key X_i at level 2 during period ℓ, the sender does the following.

1. Pick a one-time signature key pair $(ssk, svk) \xleftarrow{R} \mathcal{G}(\lambda)$ and set $C_1 = svk$.
2. Choose $r \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$C_2 = X_i^r \qquad C_3 = e(g, h_\ell)^r \cdot m \qquad C_4 = (u^{svk} \cdot v)^r$$

3. Generate a one-time signature $\sigma = \mathcal{S}(ssk, (\ell, C_3, C_4))$ on (ℓ, C_3, C_4).

The ciphertext is $C_i = (\ell, C_1, C_2, C_3, C_4, \sigma)$.

ReEnc$(R_{ij\ell}, \ell, C_i)$**:** on input of the re-encryption key $R_{ij\ell} = h_\ell^{x_j/x_i}$ and a ciphertext $C_i = (C_1, C_2, C_3, C_4, \sigma)$, the validity of the latter can be checked exactly as in section A.3 (i.e. conditions (A.1)-(A.2) must be satisfied). If ill-formed, C_i is declared '`invalid`'. Otherwise, it can be re-encrypted by choosing $t \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computing

$$C_2' = X_i^t \qquad C_2'' = R_{ij\ell}^{1/t} = h_\ell^{(x_j/x_i)t^{-1}} \qquad C_2''' = C_2^t = X_i^{rt}$$

The re-encrypted ciphertext is $C_j = (\ell, C_1, C_2', C_2'', C_2''', C_3, C_4, \sigma)$.

Dec$_1(C_j, sk_j)$**:** a first level ciphertext C_j is deemed valid if it satisfies similar conditions to (A.3)-(A.5) in the scheme of section A.3. Namely, we must have

$$e(C_2', C_2'') = e(X_j, h_\ell) \tag{A.9}$$

$$e(C_2''', u^{C_1} \cdot v) = e(C_2', C_4) \tag{A.10}$$

$$\mathcal{V}(svk, \sigma, (\ell, C_3, C_4)) = 1 \tag{A.11}$$

If C_j is valid, the plaintext $m = C_3/e(C_2'', C_2''')^{1/x_j}$ is returned. Otherwise, the message '`invalid`' is returned.

Dec$_2(C_i, sk_i)$**:** the receiver i outputs '`invalid`' if the second-level ciphertext

$$C_i = (\ell, C_1, C_2, C_3, C_4, \sigma)$$

is ill-formed. Otherwise, it outputs $m = C_3/e(C_2, h_\ell)^{1/x_i}$.

For such a scheme with temporary delegation, replayable chosen-ciphertext security can be defined by naturally extending definition A.2.2. At the beginning of each time period, the attacker obtains all honest-to-honest, corrupt-to-corrupt and corrupt-to-honest re-encryption keys. At the end of a time interval, she also receives all honest-to-corrupt re-encryption keys if she did not choose to be challenged during that period. When she decides to enter the challenge phase at some period $\ell^\star$, she obtains a challenge ciphertext as well as honest-to-corrupt keys $R_{ij\ell^\star}$ for $i \neq i^\star$.

Throughout all periods, she can access a first level decryption oracle and a re-encryption oracle that uses the current re-encryption keys. As she obtains re-encryption keys in chronological order, it is reasonable to expect that queries are made in chronological order as well. Here, a second level decryption oracle is again useless since second level ciphertexts can be publicly "sent" to the first level while keeping the plaintext and the receiver unchanged.

With this security definition, we can prove the security of this scheme under a slightly stronger (but still reasonable) assumption than in section A.3. This assumption, that we call 4-QDBDH, states that it dwells hard to recognize $e(g,g)^{b/a}$ given $(g^a, g^{(a^2)}, g^{(a^3)}, g^{(a^4)}, g^b)$. Again, this assumption is not stronger than the q-DBDHI assumption [BB04a] for $q \geq 4$.

Theorem A.4.1 Assuming the strong unforgeability of the one-time signature, the scheme is RCCA-secure at both levels under the 4-QDBDH assumption.

Proof: Detailed in the full version of the paper. ▌

A.5 Conclusions and Open Problems

We presented the first unidirectional proxy re-encryption scheme with chosen-ciphertext security in the standard model (i.e. without using the random oracle heuristic). Our construction is efficient and demands a reasonable intractability assumption in bilinear groups. In addition, we applied the same ideas to construct a chosen-ciphertext secure PRE scheme with temporary delegation.

Many open problems still remain. For instance, Canetti and Hohenberger suggested [CH07] to investigate the construction of a multi-hop unidirectional PRE system. They also mentioned the problem of securely obfuscating CCA-secure re-encryption or other key translation schemes. It would also be interesting to efficiently implement such primitives outside bilinear groups (the recent technique from [BGH07] may be useful regarding this issue). Finally, as mentioned in the end of section A.2.1, the design a scheme withstanding transfer of delegation attacks is another challenging task.

Acknowledgements

We are grateful to Jorge Villar for many useful comments and suggestions. We also thank anonymous PKC referees for their comments and Susan Hohenberger for helpful discussions on security models. The first author was supported by the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.).

Appendix B

Multi-Use Unidirectional Proxy Re-Signatures

ACM CCS 2008

[LV08a] with B. Libert

Abstract : *In 1998, Blaze, Bleumer, and Strauss suggested a cryptographic primitive termed* proxy re-signature *in which a proxy transforms a signature computed under Alice's secret key into one from Bob on the same message. The proxy is only semi-trusted in that it cannot learn any signing key or sign arbitrary messages on behalf of Alice or Bob. At CCS 2005, Ateniese and Hohenberger revisited this primitive by providing appropriate security definitions and efficient constructions in the random oracle model. Nonetheless, they left open the problem of constructing a multi-use unidirectional scheme where the proxy is only able to translate in one direction and signatures can be re-translated several times.*

This paper provides the first steps towards efficiently solving this problem, suggested for the first time 10 *years ago, and presents the first* multi-hop unidirectional *proxy re-signature schemes. Although our proposals feature a linear signature size in the number of translations, they are the first multi-use realizations of the primitive that satisfy the requirements of the Ateniese-Hohenberger security model. The first scheme is secure in the random oracle model. Using the same underlying idea, it readily extends into a secure construction in the standard model (i.e. the security proof of which avoids resorting to the random oracle idealization). Both schemes are computationally efficient but require newly defined Diffie-Hellman-like assumptions in bilinear groups.*

B.1 Introduction

In 1998, Blaze, Bleumer and Strauss [BBS98] introduced a cryptographic primitive where a semi-trusted proxy is provided with some information that allows turning Alice's signature on a message into Bob's signature on the same message. These *proxy re-signatures* (PRS) – not to be confused with proxy signatures [MUO96] – require that proxies be unable to sign on behalf of Alice or Bob on their own. The recent years saw a renewed interest of the research community in proxy re-cryptography [AFGH06, AH05, GA07, Hoh06, HRsV07, CH07].

This paper presents the first constructions of *multi-use unidirectional* proxy re-signature wherein the proxy can only translate signatures in one direction and messages can be re-signed a polynomial number of times. Our constructions are efficient and demand new (but falsifiable) Diffie-Hellman-related intractability assumptions in bilinear map groups. One of our contributions is a secure scheme in the standard model (*i.e.* without resorting to the random oracle model).

Related work. Alice – the delegator – can easily designate a proxy translating signatures computed using the secret key of Bob – the delegatee – into one that are valid w.r.t. her public key by storing her secret key at the proxy. Upon receiving Bob's signatures, the proxy can check them and re-sign the message using Alice's private key. The problem with this approach is that the proxy can sign arbitrary messages on behalf of Alice. Proxy re-signatures aim at securely enabling the delegation of signatures without fully trusting the proxy. They are related to proxy signatures [MUO96,ID03] in that any PRS can be used to implement a proxy signature mechanism but the converse is not necessarily true.

In 1998, Blaze *et al.* [BBS98] gave the first example of PRS where signing keys remain hidden from the proxy. The primitive was formalized in 2005 by Ateniese and Hohenberger [AH05] who pinned down useful properties that can be expected from proxy re-signature schemes:

- Unidirectionality: re-signature keys can only be used for delegation in one direction;
- Multi-usability: a message can be re-signed a polynomial number of times;
- Privacy of proxy keys: re-signature keys can be kept secret by honest proxies;
- Transparency: users may not even know that a proxy exists;
- Unlinkability: a re-signature cannot be linked to the signature from which it was generated;
- Key optimality: a user is only required to store a constant amount of secret data;
- Non-interactivity: the delegatee does not act in the delegation process;
- Non-transitivity: proxies cannot re-delegate their re-signing rights.

Blaze *et al.*'s construction is *bidirectional* (*i.e.* the proxy information allows "translating" signatures in either direction) and *multi-use* (*i.e.* the translation of signatures can be performed in sequence and multiple times by distinct proxies without requiring the intervention of signing entities). Unfortunately, Ateniese and Hohenberger [AH05] pinpointed a flaw in the latter scheme: given a signature/re-signature pair, anyone can deduce the re-signature key that has been used in the delegation (*i.e.* proxy keys are not private). Another issue in [BBS98] is that the proxy and the delegatee can collude to expose the delegator's secret.

To overcome these limitations, Ateniese and Hohenberger [AH05] proposed two constructions based on bilinear maps. The first one is a multi-use, bidirectional extension of Boneh-Lynn-Shacham (BLS) signatures [BLS04]. Their second scheme is unidirectional (the design of such a scheme was an open problem raised in [BBS98]) but single-use. It involves two different signature algorithms: *first-level* signatures can be translated by the proxy whilst *second-level* signatures (that are obtained by translating first level ones or by signing at level 2) cannot. A slightly less efficient variant was also suggested to ensure the privacy of re-signature keys kept at the proxy. The security of all schemes was analyzed in the random oracle model [BR93].

Motivations. A number of applications were suggested in [AH05] to motivate the search for unidirectional systems. One of them was to provide a proof that a certain path was taken in a directed graph: to make sure that a foreign visitor legally entered the country and went through

the required checkpoints, U.S. customs only need one public key (the one of the immigration service once the original signature on the e-passport has been translated by an immigration agent). Optionally, the final signature can hide which specific path was chosen and only vouch for the fact that an authorized one was taken. In such a setting, proxy re-signatures are especially interesting when they are multi-use.

Another application was the sharing and the conversion of digital certificates: valid signatures for untrusted public keys can be turned into signatures that verify under already certified keys so as to save the cost of obtaining a new certificate. As exemplified in [AH05], unidirectional schemes are quite appealing for converting certificates between *ad-hoc* networks: using the public key of network B's certification authority (CA), the CA of network A can non-interactively compute a translation key and set up a proxy converting certificates from network B within its own domain without having to rely on untrusted nodes of B.

As a third application, PRS can be used to implement anonymizable signatures that hide the internal organization of a company. Outgoing documents are first signed by specific employees. Before releasing them to the outside world, a proxy translates signatures into ones that verify under a corporate public key so as to conceal the original issuer's identity and the internal structure of the company.

Our contributions. Ateniese and Hohenberger left as open challenges the design of multi-use unidirectional systems and that of secure schemes in the standard security model. This paper provides solutions to both problems:

- we present a simple and efficient system (built on the short signature put forth by Boneh *et al.* [BLS04]) which is secure in the random oracle model under an appropriate extension of the Diffie-Hellman assumption;
- using an elegant technique due to Waters [Wat05], the scheme is easily modified so as to achieve security in the standard model. To the best of our knowledge, this actually provides the first unidirectional PRS that dispenses with random oracles and thereby improves a recent bidirectional construction [SCWL07].

Both proposals additionally preserve the privacy of proxy keys (with an improved efficiency w.r.t. [AH05] in the case of the first one). They combine almost all of the above properties. As in prior unidirectional schemes, proxies are not completely transparent since signatures have different shapes and lengths across successive levels. The size of our signatures actually grows linearly with the number of past translations: signatures at level ℓ (*i.e.* that have been translated $\ell - i$ times if the original version was signed at level i) consist of about 2ℓ group elements. In spite of this blow-up, we retain important benefits:

- signers may tolerate a limited number (say t) of signature translations for specific messages. Then, if L distinct signature levels are permitted in the global system, users can directly sign messages at level $L - t$.
- the conversion of a ℓ^{th} level signature is indistinguishable from one generated at level $\ell + 1$ by the second signer. The original signer's identity is moreover perfectly hidden and the verifier only needs the new signer's public key.

As a last contribution, we also show how the single-hop restrictions of both schemes can be modified in such a way that one can prove their security in the stronger *plain public key model* (also considered in [BN06] for different primitives). Prior works on proxy re-cryptography consider security definitions where dishonest parties' public keys are honestly generated and the corresponding secret key is known to the attacker. Relying on the latter assumption requires CAs to ask for a proof of knowledge of the associated private key before certifying a public key.

As exemplified in [BN06], not all security infrastructures do rigorously apply such an advisable practice. To address this issue in our setting, we extend the security definitions of [AH05] to the *plain public key model* (a.k.a. *chosen-key model*) where the adversary is allowed to choose public keys on behalf of corrupt users (possibly non-uniformly or as a function of honest parties' public keys) without being required to reveal or prove knowledge of the underlying private key. In our model, we are able to construct single-hop unidirectional schemes that are secure in the plain public key model. The practical impact of this result is that users do not have to demonstrate knowledge of their secret upon certification. They must only obtain a standard certificate such as those provided by current PKIs.

Organization. In the forthcoming sections, we recall the syntax of unidirectional PRS schemes and the security model in section B.2. Section B.3 explains which algorithmic assumptions we need. Section B.4 describes our random-oracle-using scheme. In section B.5, we detail how to get rid of the random oracle idealization. Section B.6 then suggests single-hop constructions in the chosen-key model.

B.2 Model and Security Notions

We first recall the syntactic definition of unidirectional PRS schemes from [AH05].

Definition B.2.1 [Proxy Re-Signatures] A (unidirectional) proxy re-signature (PRS) scheme for N signers and L levels (where N and L are both polynomial in the security parameter λ) is a tuple of (possibly randomized) algorithms ($\mathsf{Global\text{-}Setup}$, Keygen, $\mathsf{ReKeygen}$, Sign, $\mathsf{Re\text{-}Sign}$, Verify) where:

$\mathsf{Global\text{-}Setup}(\lambda)$**:** is a randomized algorithm (possibly run by a trusted party) that takes as input a security parameter λ and produces system-wide public parameters cp.

$\mathsf{Keygen}(\mathsf{cp})$**:** is a probabilistic algorithm that, on input of public parameters cp, outputs a signer's private/public key pair (sk, pk).

$\mathsf{ReKeygen}(\mathsf{cp}, pk_i, sk_j)$**:** on input of public parameters cp, the public key pk_i of signer i and signer j's private key sk_j, this (possibly randomized but ideally non-interactive) algorithm outputs a re-signature key R_{ij} that allows turning i's signatures into signatures in the name of j.

$\mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m)$**:** on input of public parameters cp, a message m, a private key sk_i and an integer $\ell \in \{1, \ldots, L\}$, this (possibly probabilistic) algorithm outputs a signature σ on behalf of signer i at level ℓ.

$\mathsf{Re\text{-}Sign}(\mathsf{cp}, \ell, m, \sigma, R_{ij}, pk_i, pk_j)$**:** given common parameters cp, a level $\ell < L$ signature σ from signer $i \in \{1, \ldots, N\}$ and a re-signature key R_{ij}, this (possibly randomized) algorithm first checks that σ is valid w.r.t pk_i. If yes, it outputs a signature σ' that verifies at level $\ell + 1$ under the public key pk_j.

$\mathsf{Verify}(\mathsf{cp}, \ell, m, \sigma, pk_i)$**:** given public parameters cp, an integer $\ell \in \{1, \ldots, L\}$, a message m, an alleged signature σ and a public key pk_i, this deterministic algorithm outputs 0 or 1.

For all security parameters $\lambda \in \mathbb{N}$ and public parameters cp output by $\mathsf{Global\text{-}Setup}(\lambda)$, for all couples of private/public key pairs (sk_i, pk_i), (sk_j, pk_j) produced by $\mathsf{Keygen}(\mathsf{cp})$, for any

$\ell \in \{1, \ldots, L\}$ and message m, we should have

$$\mathsf{Verify}(\mathsf{cp}, \ell, m, \mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m), pk_i) = 1;$$
$$\mathsf{Verify}(\mathsf{cp}, \ell + 1, m, \sigma, pk_j) = 1.$$

whenever $\sigma = \mathsf{ReSign}(\mathsf{cp}, \ell, m, \mathsf{Sign}(\mathsf{cp}, \ell, sk_i, m), R_{ij})$ and $R_{ij} = \mathsf{ReKeygen}(\mathsf{cp}, pk_i, sk_j)$.

To lighten notations, we sometimes omit to explicitly include public parameters cp that are part of the input of some of the above algorithms.

The security model of [AH05] considers the following two orthogonal notions termed *external* and *insider security*.

External security: is the security against adversaries outside the system (that differ from the proxy and delegation partners). This notion demands that the next probability be a negligible function of the security parameter λ:

$$\Pr[\quad \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[1,N]}, (i^\star, L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(.),\mathcal{O}_{Resign}(.)}(\{pk_i\}_{i\in[1,N]}) :$$
$$\mathsf{Verify}(L, m^\star, \sigma^\star, pk_{i^\star}) \wedge (i^\star, m^\star) \notin Q]$$

where $\mathcal{O}_{Sign}(.)$ is an oracle taking as input a message and an index $i \in \{1, \ldots, N\}$ to return a 1^{st}-level signature $\sigma \leftarrow \mathsf{Sign}(1, sk_i, m)$; $\mathcal{O}_{Resign}(.)$ takes indices $i, j \in \{1, \ldots, N\}$ and a ℓ^{th}-level signature σ to output $\sigma' \leftarrow \mathsf{Re\text{-}Sign}(\ell, m, \sigma, \mathsf{ReKeygen}(pk_i, sk_j), pk_i, pk_j)$; and Q denotes the set of (signer,message) pairs (i, m) queried to $\mathcal{O}_{Sign}(.)$ or such that a tuple $(?, j, i, m)$, with $j \in \{1, \ldots, N\}$, was queried to $\mathcal{O}_{Resign}(.)$. This notion only makes sense if re-signing keys are kept private by the proxy.

In our setting, the translation of a ℓ^{th}-level signature is perfectly indistinguishable from a signature produced by the delegator at level $\ell + 1$. Therefore, we can always simulate the $\mathcal{O}_{Resign}(.)$ oracle by publicly "sending" outputs of $\mathcal{O}_{Sign}(.)$ to the next levels. For the sake of generality, we nevertheless leave $\mathcal{O}_{Resign}(.)$ in the definition.

Internal security: The second security notion considered in [AH05] strives to protect users against dishonest proxies and colluding delegation partners. Three security guarantees should be ensured.

1. **Limited Proxy security:** this notion captures the proxy's inability to sign messages on behalf of the delegatee or to create signatures for the delegator unless messages were first signed by one of the latter's delegatees. Formally, we consider a game where adversaries have all re-signing keys but are denied access to signers' private keys. The following probability should be negligible:

$$\Pr[\quad \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[1,N]}, \{R_{ij} \leftarrow \mathsf{ReKeygen}(pk_i, sk_j)\}_{i,j\in[1,N]},$$
$$(i^\star, L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(.,.)}(\{pk_i\}_{i\in[1,N]}, \{R_{ij}\}_{i,j\in[1,N]}) :$$
$$\mathsf{Verify}(L, m^\star, \sigma^\star, pk_{i^\star}) \wedge m^\star \notin Q]$$

where $\mathcal{O}_{Sign}(.,.)$ is an oracle taking as input a message and an index $i \in \{1, \ldots, N\}$ to return a first level signature $\sigma \leftarrow \mathsf{Sign}(1, sk_i, m)$ and Q stands for the set of messages m queried to the signing oracle.

2. **Delegatee Security:** informally, this notion protects the delegatee from a colluding delegator and proxy. Namely, the delegatee is assigned the index 0. The adversary is provided with an oracle returning first level signatures on behalf of 0. Knowing corrupt users' private keys, she can compute re-signature keys $\{R_{ij}\}_{i\in\{0,\ldots,N\},j\in\{1,\ldots,N\}}$ on her own[1] from pk_i and sk_j, with $j \neq 0$. Obviously, she is not granted access to R_{i0} for any $i \neq 0$. Her probability of success

$$\Pr\Big[\quad \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[0,N]}, \\ (L, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(0,.)}(pk_0, \{pk_i, sk_i\}_{i\in[1,N]}) : \\ \mathsf{Verify}(L, m^\star, \sigma^\star, pk_0) \wedge m^\star \notin Q \Big],$$

   where Q is the set of messages queried to $\mathcal{O}_{Sign}(0,.)$, should be negligible.

3. **Delegator Security:** this notion captures that a collusion between the delegatee and the proxy should be harmless for the honest delegator. More precisely, we consider a target delegator with index 0. The adversary is given private keys of all other signers $i \in \{1,\ldots,N\}$ as well as *all* re-signature keys including R_{i0} and R_{0i} for $i \in \{1,\ldots,N\}$. A signing oracle $\mathcal{O}_{Sign}(0,.)$ also provides her with first level signatures for 0. Yet, the following probability should be negligible,

$$\Pr\Big[\quad \{(pk_i, sk_i) \leftarrow \mathsf{Keygen}(\lambda)\}_{i\in[0,N]}, \{R_{ij} \leftarrow \mathsf{ReKeygen}(pk_i, sk_j)\}_{i,j\in[0,N]}, \\ (1, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(0,.)}(pk_0, \{pk_i, sk_i\}_{i\in[1,N]}, \{R_{ij}\}_{i,j\in[0,N]},) : \\ \mathsf{Verify}(1, m^\star, \sigma^\star, pk_0) \wedge m^\star \notin Q\Big],$$

   meaning she has little chance of framing user 0 at the first level.

An important difference between external and limited proxy security should be underlined. In the former, the attacker is allowed to obtain signatures on the target message $m^\star$ for signers other than $i^\star$. In the latter, the target message cannot be queried for signature at all (knowing all proxy keys, the attacker would trivially win the game otherwise).

Chosen-Key Model Security. As in other papers on proxy re-cryptography [AFGH06, CH07], the above model assumes that users only publicize a public key if they hold the underlying private key. This actually amounts to use a trusted key generation model or the so-called *knowledge-of-secret-key* model (KOSK), introduced in [Bol03], that demands attackers to reveal the associated private key whenever they create a public key for themselves. This model (sometimes referred to as the *registered key model*) mirrors the fact that, in a PKI, users should prove knowledge of their private key upon certification of their public key.

As argued by Bellare and Neven in a different context [BN06], relying on the registered key model can be quite burdensome in real world applications if one is willing to actually implement the requirements of that model. Although some kinds of proof of private key possession [RY07] are implemented by VeriSign and other security infrastructures, they are far from sufficing to satisfy assumptions that are implicitly made by the KOSK model. To do so, CAs should implement complex proofs of knowledge that allow for the online extraction of adversarial secrets so as to remain secure in a concurrent setting like the Internet, where many users may be willing to register at the same time. Hence, whenever it is possible, one should preferably work in a

[1] This is true in in non-interactive schemes, which we are focusing on. In the general case, those keys should be generated by the challenger and explicitly provided as input to the adversary.

model called *chosen-key model* (a.k.a. *plain public key model*) that leaves adversaries choose their public key as they like (possibly as a function of honest parties' public keys and without having to know or reveal the underlying secret whatsoever).

If we place ourselves in the chosen-key model, the notions of external security and limited proxy security are not altered as they do not involve corrupt users. On the other hand, we need to recast the definitions of delegatee and delegator security and take adversarially-generated public keys into account. As to the delegatee security, the only modification is that the adversary is challenged on a single public key. No other change is needed since $\mathcal{A}$ can generate re-signature keys on her own. In the notion of delegator security, $\mathcal{A}$ is also challenged on a single public key pk_0 for which she is granted access to a first level signing oracle. In addition, we introduce a delegation oracle $\mathcal{O}_{dlg}(.)$ that delegates on behalf of user 0. When queried on a public key pk_i supplied by the adversary, $\mathcal{O}_{dlg}(.)$ responds with $R_{i0} = \mathsf{ReKeygen}(pk_i, sk_0)$.

We stress that we are not claiming that the schemes of [AH05] are insecure in such a model. However, their security is not guaranteed any longer with currently known security proofs. In section B.6, we will explain how to simply modify the single-hop versions of our schemes so as to prove them secure without making the KOSK assumption.

B.3 Bilinear Maps and Complexity Assumptions

Bilinear groups. Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is an efficiently computable mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with these properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

Flexible Diffie-Hellman problems. Our signatures rely on new generalizations of the Computational Diffie-Hellman (CDH) problem which is to compute g^{ab} given (g^a, g^b) in a group $\mathbb{G} = \langle g \rangle$. To motivate them, let us first recall the definition of the *2-out-of-3 Diffie-Hellman* problem [KJP06].

Definition B.3.1 In a prime order group $\mathbb{G}$, the **2-out-of-3 Diffie-Hellman** problem (2-3-CDH) is, given (g, g^a, g^b), to find a pair $(C, C^{ab}) \in \mathbb{G} \times \mathbb{G}$ with $C \neq 1_{\mathbb{G}}$.

We introduce a potentially harder version of this problem that we call 1-Flexible Diffie-Hellman problem:

Definition B.3.2 The **1-Flexible Diffie-Hellman** problem (1-FlexDH) is, given $(g, g^a, g^b) \in \mathbb{G}^3$, to find a triple $(C, C^a, C^{ab}) \in (\mathbb{G} \backslash \{1_{\mathbb{G}}\})^3$.

We shall rely on a relaxed variant of this problem where more flexibility is permitted in the choice of the base C for the Diffie-Hellman computation.

Definition B.3.3 The **ℓ-Flexible Diffie-Hellman** problem (ℓ-FlexDH) (for $\ell \geq 1$) is, given a triple $(g, g^a, g^b) \in \mathbb{G}^3$, to find a $(2\ell + 1)$-tuple

$$(C_1, \ldots, C_\ell, D_1^a, \ldots, D_\ell^a, D_\ell^{ab}) \in \mathbb{G}^{2\ell+1}$$

where $\log_g(D_j) = \prod_{i=1}^{j} \log_g(C_i) \neq 0$ for $j \in \{1, \ldots \ell\}$.

A given instance has many publicly verifiable solutions: a candidate $2\ell + 1$-tuple

$$(C_1, \ldots, C_\ell, D'_1, \ldots, D'_\ell, T)$$

is acceptable if $e(C_1, A) = e(D'_1, g)$, $e(D'_j, g) = e(D'_{j-1}, C_j)$ for $j = 2, \ldots, \ell$ and $e(D'_\ell, B) = e(T, g)$. The ℓ-FlexDH assumption is thus falsifiable according to Naor's classification [Nao03].

In generic groups, the general intractability result given by theorem 1 of [KJP06] by Kunz-Jacques and Pointcheval implies the generic hardness of ℓ-FlexDH. Section B.8 gives an adaptation of this result in generic *bilinear* groups.

Remark The *knowledge-of-exponent assumption* (KEA1) [BP04a] was introduced by Damgård [Dam91]. Roughly speaking, it captures the intuition that any algorithm $\mathcal{A}$ which, given elements (g, g^x) in $\mathbb{G}^2$, computes a pair $(h, h^x) \in \mathbb{G}^2$ must "know" $\log_g(h)$. Hence, it must be feasible to recover the latter value using $\mathcal{A}$'s random coins. In [BP04b], Bellare and Palacio defined a slightly stronger variant (dubbed DHK1 as a shorthand for "Diffie-Hellman knowledge") of this assumption. DHK1 essentially says that, given a pair (g, g^x), for any adversary $\mathcal{A}$ that outputs pairs (h_i, h_i^x), there exists an extractor that can always recover $\log_g(h_i)$ using $\mathcal{A}$'s random coins. The latter is allowed to query the extractor on polynomially-many pairs (h_i, h_i^x). For each query, $\mathcal{A}$ first obtains $\log_g(h_i)$ from the extractor before issuing the next query. Under DHK1, the intractability of the ℓ-Flexible Diffie-Hellman problem is easily seen to boil down to the Diffie-Hellman assumption. Given a pair (g, g^a), a polynomial adversary that outputs $(C_1, D_1^a) = (C_1, C_1^a)$ necessarily "knows" $t_1 = \log_g C_1$ and thus also $(C_2, C_2^a) = (C_2, (D_2^a)^{1/t_1})$ as well as $t_2 = \log_g C_2$, which in turn successively yields logarithms of $C_3, \ldots, C_\ell$. Although DHK1-like assumptions are inherently non-falsifiable, they hold in generic groups [Den06, AF07] and our results can be seen as resting on the combination CDH+DHK1.

Modified Diffie-Hellman problem. The second assumption that we need is that the CDH problem (g^a, g^b) remains hard even when $g^{(a^2)}$ is available.

Definition B.3.4 The **modified Computational Diffie-Hellman** problem (mCDH) is, given $(g, g^a, g^{(a^2)}, g^b) \in \mathbb{G}^4$, to compute $g^{ab} \in \mathbb{G}$.

In fact, we use an equivalent formulation of the problem which is to find h^{xy} given $(h, h^x, h^{1/x}, h^y)$ (where we set $g = h^{1/x}$, $x = a$, $y = b/a$).

B.4 A Multi-Hop Scheme in the Random Oracle Model

To provide a better intuition of the underlying idea of our scheme, we first describe its single-hop version before extending it into a multi-hop system.

Our approach slightly differs from the one in [AH05] where signers have a "strong" secret and a "weak" secret that are respectively used to produce first and second level signatures. In our scheme, users have a single secret but first and second level signatures retain different shapes. Another difference is that our re-signature algorithm is probabilistic.

We exploit the idea that, given $g^b \in \mathbb{G} = \langle g \rangle$ for some $b \in \mathbb{Z}$, one can hardly generate a Diffie-Hellman triple (g^a, g^b, g^{ab}) without knowing the corresponding exponent a [Dam91]. A valid BLS signature [BLS04] $(\sigma = H(m)^x, X = g^x)$ can be blinded into $(\sigma'_1, \sigma'_2) = (\sigma^t, X^t)$ using a random exponent t. An extra element g^t then serves as evidence that (σ'_1, σ'_2) actually hides a valid pair. This technique can be iterated several times by adding two group elements at each step. To translate signatures from signer i to signer j, the key idea is to have the proxy perform an appropriate change of variable involving the translation key during the blinding.

The scheme is obviously not strongly unforgeable in the sense of [ADR02] (since all but first level signatures can be publicly re-randomized) but this "malleability" of signatures is not a weakness whatsoever. It even turns out to be a desirable feature allowing for the unlinkability of translated signatures w.r.t. original ones.

B.4.1 The Single Hop Version

In this scheme, signers' public keys consist of a single group element $X = g^x \in \mathbb{G}$. Their well-formedness is thus efficiently verifiable by the certification authority that just has to check their membership in $\mathbb{G}$. This already improves [AH05] where public keys $(X_1, X_2) = (g^x, h^{1/x}) \in \mathbb{G}^2$ (g and h being common parameters) must be validated by testing whether $e(X_1, X_2) = e(g, h)$.

Global-setup(λ)**:** this algorithm chooses bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. A generator $g \in \mathbb{G}$ and a hash function $H : \{0,1\}^* \to \mathbb{G}$ (modeled as a random oracle in the security proof) are also chosen. Public parameters only consist of $\mathsf{cp} := \{\mathbb{G}, \mathbb{G}_T, g, H\}$.

Keygen(λ)**:** user i's public key is set as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

ReKeygen(x_j, X_i)**:** this algorithm outputs the re-signature key $R_{ij} = X_i^{1/x_j} = g^{x_i/x_j}$ which allows turning signatures from i into signatures from j.

Sign$(1, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 1, compute $\sigma^{(1)} = H(m)^{x_i} \in \mathbb{G}$.

Sign$(2, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 2, choose $t \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2) = (H(m)^{x_i t}, X_i^t, g^t). \tag{B.1}$$

Re-Sign$(1, m, \sigma^{(1)}, R_{ij}, X_i, X_j)$**:** on input of $m \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a signature $\sigma^{(1)} \in \mathbb{G}$ and public keys X_i, X_j, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing $e(\sigma^{(1)}, g) = e(H(m), X_i)$. If valid, $\sigma^{(1)}$ is turned into a signature on behalf of j by choosing $t \xleftarrow{R} \mathbb{Z}_p^*$ and computing

$$\sigma^{(2)} = (\sigma'_0, \sigma'_1, \sigma'_2) \quad = \quad ({\sigma^{(1)}}^t, X_i^t, R_{ij}^t) = (H(m)^{x_i t}, X_i^t, g^{t x_i/x_j})$$

If we set $\tilde{t} = t x_i/x_j$, we have

$$\sigma^{(2)} = (\sigma'_0, \sigma'_1, \sigma'_2) = (H(m)^{x_j \tilde{t}}, X_j^{\tilde{t}}, g^{\tilde{t}}). \tag{B.2}$$

Verify$(1, m, \sigma^{(1)}, X_i)$**:** accept $\sigma^{(1)}$ if $e(\sigma^{(1)}, g) = e(H(m), X_i)$.

Verify$(2, m, \sigma^{(2)}, X_i)$**:** a 2^{nd} level signature $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2)$ is accepted for the public key X_i if the following conditions are true.

$$e(\sigma_0, g) = e(\sigma_1, H(m)) \qquad e(\sigma_1, g) = e(X_i, \sigma_2)$$

Relations (B.1) and (B.2) show that translated signatures have exactly the same distribution as signatures directly produced by signers at level 2.

In comparison with the only known unidirectional PRS with private re-signing keys (suggested in section 3.4.2 of [AH05]), this one features shorter second level signatures that must include a Schnorr-like [Sch91] proof of knowledge in addition to 3 group elements in [AH05]. On the other hand, signatures of [AH05] are strongly unforgeable unlike ours.

It is also worth mentioning that the above scheme only requires the 1-Flexible Diffie-Hellman assumption which is more classical than the general ℓ-FlexDH.

B.4.2 How to Obtain Multiple Hops

The above construction can be scaled up into a multi-hop PRS scheme if we iteratively apply the same idea several times. To prevent the linkability of signatures between successive levels $\ell+1$ and $\ell+2$, the re-signature algorithm performs a re-randomization using random exponents $r_1, \ldots, r_\ell$.

Sign$(\ell+1, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at the $(\ell+1)^{\text{th}}$ level, user i chooses $(t_1, \ldots, t_\ell) \xleftarrow{R} (\mathbb{Z}_p^*)^\ell$ and outputs $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell}) \in \mathbb{G}^{2\ell+1}$ where

$$\begin{cases} \sigma_0 = H(m)^{x_i t_1 \cdots t_\ell} & \\ \sigma_k = g^{x_i t_1 \cdots t_{\ell+1-k}} & \text{for } k \in \{1, \ldots, \ell\} \\ \sigma_k = g^{t_{k-\ell}} & \text{for } k \in \{\ell+1, \ldots, 2\ell\}. \end{cases}$$

Re-Sign$(\ell+1, m, \sigma^{(\ell+1)}, R_{ij}, X_i, X_j)$**:** on input of a message $m \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a valid $(\ell+1)^{\text{th}}$-level signature

$$\begin{aligned} \sigma^{(\ell+1)} &= (\sigma_0, \ldots, \sigma_{2\ell}) \\ &= (H(m)^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_{\ell-1}}, \ldots, g^{x_i t_1}, g^{t_1}, \ldots, g^{t_\ell}) \in \mathbb{G}^{2\ell+1} \end{aligned}$$

and public keys X_i, X_j, check the validity of $\sigma^{(\ell+1)}$ under X_i. If valid, it is turned into a $(\ell+2)^{\text{th}}$-level signature on behalf of j by drawing $(r_0, \ldots, r_\ell) \xleftarrow{R} (\mathbb{Z}_p^*)^{\ell+1}$ and computing $\sigma^{(\ell+2)} = (\sigma'_0, \ldots, \sigma'_{2\ell+2}) \in \mathbb{G}^{2\ell+3}$ where

$$\begin{cases} \sigma'_0 = \sigma_0^{r_0 \cdots r_\ell} & \\ \sigma'_k = \sigma_k^{r_0 \cdots r_{\ell+1-k}} & \text{for } k \in \{1, \ldots, \ell\} \\ \sigma'_{\ell+1} = X_i^{r_0} & \\ \sigma'_{\ell+2} = R_{ij}^{r_0} & \\ \sigma'_k = \sigma_{k-2}^{r_{k-\ell-2}} & \text{for } k \in \{\ell+3, \ldots, 2\ell+2\}. \end{cases}$$

If we define $\tilde{t}_0 = r_0 x_i / x_j$ and $\tilde{t}_k = r_k t_k$ for $k = 1, \ldots, \ell$, we observe that

$$\sigma^{(\ell+2)} = (H(m)^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_{\ell-1}}, \ldots, g^{x_j \tilde{t}_0}, g^{\tilde{t}_0}, \ldots, g^{\tilde{t}_\ell}) \in \mathbb{G}^{2\ell+3}$$

Verify$(\ell+1, m, \sigma^{(\ell+1)}, X_i)$**:** at level $(\ell+1)$, the validity of $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell}) \in \mathbb{G}^{2\ell+1}$ is checked by testing if these equalities simultaneously hold:

$$\begin{aligned} e(\sigma_0, g) &= e(H(m), \sigma_1), \\ e(\sigma_\ell, g) &= e(X_i, \sigma_{\ell+1}) \\ e(\sigma_k, g) &= e(\sigma_{k+1}, \sigma_{2\ell-k+1}) \text{ for } k \in \{1, \ldots, \ell-1\} \end{aligned}$$

We note that the speed of the verification algorithm can be increased by computing a product of $O(\ell)$ pairings, which is significantly faster than $O(\ell)$ independent pairing calculations [GS06]. The idea is to choose $\omega_0, \ldots, \omega_\ell \xleftarrow{R} \mathbb{Z}_p^*$ at random and check whether

$$e(g, \prod_{k=0}^{\ell} \sigma_k^{\omega_k}) = e(H(m), {\sigma_1}^{\omega_0}) \cdot e(X_i, \sigma_{\ell+1}^{\omega_\ell}) \cdot \prod_{k=1}^{\ell-1} e(\sigma_{k+1}, \sigma_{2\ell-k+1}^{\omega_k}).$$

With high probability, invalid signatures fail to satisfy the above randomized verification algorithm.

B.4.3 Security

Theorem B.4.1 The L-level scheme is a secure unidirectional proxy re-signature under the $(L-1)$-FlexDH and mCDH assumptions in the random oracle model.

Proof: We first prove security against dishonest proxies.

Limited proxy security From an adversary $\mathcal{A}_1$ with advantage ε, we can construct an algorithm $\mathcal{B}_1$ that solves a $(L-1)$-FlexDH instance $(g, A = g^a, B = g^b)$ with probability $O(\varepsilon/q_s)$, where q_s is the number of signing queries.

System parameters: $\mathcal{A}_1$ is challenged on public parameters $\{\mathbb{G}, \mathbb{G}_T, g, \mathcal{O}_H\}$ where $\mathcal{O}_H$ is the random oracle controlled by the simulator $\mathcal{B}_1$.

Public key generation: when $\mathcal{A}_1$ asks for the creation of user $i \in \{1, \ldots, N\}$, $\mathcal{B}_1$ responds with a newly generated public key $X_i = A^{x_i} = g^{ax_i}$, for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$, which virtually defines user i's private key as ax_i. For all pairs (i, j), re-signature keys R_{ij} are calculated as $R_{ij} = g^{x_i/x_j} = g^{ax_i/ax_j}$.

Oracle queries: $\mathcal{A}_1$'s queries are tackled with as follows. Following a well-known technique due to Coron [Cor00], a binary coin $c \in \{0, 1\}$ with expected value $1 - \zeta \in [0, 1]$ decides whether $\mathcal{B}_1$ introduces the challenge in the output of the random oracle or an element of known signature. For the optimal value of ζ, this introduces the loss factor $O(q_s)$ in the success probability.

- *Random oracle queries*: to answer these queries, $\mathcal{B}_1$ maintains a list (referred to as the H-List) of tuples (m, h, μ, c) as follows:
  1. If the query m already appears in the H-List, then $\mathcal{B}_1$ returns h;
  2. Otherwise, $\mathcal{B}_1$ generates a random bit c such that $\Pr[c = 0] = \zeta$;
  3. It picks $\mu \xleftarrow{R} \mathbb{Z}_p^*$ at random and computes $h = g^\mu$ if $c = 0$ and $h = B^\mu$ otherwise;
  4. It adds the 4-uple (m, h, μ, c) to the H-List and returns h as the random oracle output.
- *Signing queries*: when a signature of signer i is queried for a message m, $\mathcal{B}_1$ runs the random oracle to obtain the 4-uple (m, h, μ, c) contained in the H-List. If $c = 1$ then $\mathcal{B}_1$ reports failure and aborts. Otherwise, the algorithm $\mathcal{B}_1$ returns $h^{x_i a} = A^{x_i \mu}$ as a valid signature on m.

After a number of queries, $\mathcal{A}_1$ comes up with a message $m^\star$, that was never queried for signature for any signer, an index $i^\star \in \{1, \ldots, N\}$ and a L^{th} level forgery $\sigma^{\star(L)} = (\sigma_0^\star, \ldots, \sigma_{2L-2}{}^\star) \in \mathbb{G}^{2L-1}$. At this stage, $\mathcal{B}_1$ runs the random oracle to obtain the 4-uple $(m^\star, h^\star, \mu^\star, c^\star)$ contained in the H-List and fails if $c^\star = 0$. Otherwise, if $\sigma^{\star(L)}$ is valid, it may be written

$$(\sigma_0^\star, \ldots, \sigma_{2L-2}{}^\star) = \left(B^{\mu^\star x_{i^\star} a t_1 \ldots t_{L-1}}, A^{t_1 \ldots t_{L-1}}, \ldots, A^{t_1}, g^{t_1}, \ldots, g^{t_{L-1}}\right)$$

which provides $\mathcal{B}_1$ with a valid tuple

$$(C_1, \ldots, C_{L-1}, D_1^a, \ldots, D_{L-1}^a, D_{L-1}^{ab}),$$

where $D_{L-1}^{ab} = \sigma_0^{\star 1/\mu^\star x_{i^\star}}$, so that $\log_g(D_j) = \prod_{i=1}^{j} \log_g(C_i)$ for $j \in \{1, \ldots, L-1\}$. A similar analysis to [Cor00, BLS04] gives the announced bound on $\mathcal{B}_1$'s advantage if the optimal probability $\zeta = q_s/(q_s + 1)$ is used when answering hash queries.

Delegatee security We also attack the $(L-1)$-FlexDH assumption using a delegatee security adversary $\mathcal{A}_2$. Given an input pair $(A = g^a, B = g^b)$, the simulator $\mathcal{B}_2$ proceeds as $\mathcal{B}_1$ did in the proof of limited proxy security.

System parameters and public keys: the target delegatee's public key is $X_0 = A = g^a$. For $i = 1, \ldots, n$, other public keys are set as $X_i = g^{x_i}$ with $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

Queries: $\mathcal{A}_2$'s hash and signing queries are handled exactly as in the proof of limited proxy security. Namely, $\mathcal{B}_2$ fails if $\mathcal{A}_2$ asks for a signature on a message m for which $H(m) = B^\mu$ and responds consistently otherwise.

When $\mathcal{A}_2$ outputs her forgery $\sigma^{\star(L)} = (\sigma_0^\star, \ldots, \sigma_{2L-2}^\star)$ at level L, $\mathcal{B}_2$ is successful if $H(m^\star) = B^{\mu^\star}$, for some $\mu^\star \in \mathbb{Z}_p^*$, and extracts an admissible $(2L-1)$-uple as done in the proof of limited proxy security.

Delegator security This security property is proven under the mCDH assumption. Given an adversary $\mathcal{A}_3$ with advantage ε, we outline an algorithm $\mathcal{B}_3$ that has probability $O(\varepsilon/q_s)$ of finding g^{ab} given $(g, A = g^a, A' = g^{1/a}, B = g^b)$.

Public key generation: as previously, the target public key is defined as $X_0 = A = g^a$. Remaining public keys are set as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$ for $i = 1, \ldots, n$. This time, $\mathcal{A}_3$ aims at producing a first level forgery and is granted *all* re-signature keys, including R_{0j} and R_{j0}. For indexes (i, j) s.t. $i, j \neq 0$, $\mathcal{B}_3$ sets $R_{ij} = g^{x_i/x_j}$. If $i = 0$, it calculates $R_{0j} = A^{1/x_j} = g^{a/x_j}$. If $j = 0$ (and thus $i \neq 0$), $\mathcal{B}_3$ computes $R_{i0} = A'^{x_i} = g^{x_i/a}$ to $\mathcal{A}_3$.

$\mathcal{A}_3$'s queries are dealt with exactly as for previous adversaries. Eventually, $\mathcal{A}_3$ produces a first level forgery $\sigma^{\star(1)}$ for a new message $m^\star$. Then, $\mathcal{B}_3$ can extract g^{ab} if $H(m) = (g^b)^{\mu^\star}$ for some $\mu^\star \in \mathbb{Z}_p^*$, which occurs with probability $O(1/q_s)$ using Coron's technique [Cor00]. Otherwise, $\mathcal{B}_3$ fails.

External security We finally show that an external security adversary $\mathcal{A}_4$ also allows breaking the $(L-1)$-FlexDH assumption almost exactly as in the proof of limited proxy security. The simulator $\mathcal{B}_4$ is given an instance $(g, A = g^a, B = g^b)$. As previously, $\mathcal{B}_4$ must "program" the random oracle H hoping that its output will be $H(m^\star) = B^{\mu^\star}$ (where $\mu^\star \in \mathbb{Z}_p^*$ is known) for the message $m^\star$ that the forgery $\sigma^{\star(L)}$ pertains to. The difficulty is that $\mathcal{B}_4$ must also be able to answer signing queries made on $m^\star$ for all but one signers. Therefore, $\mathcal{B}_4$ must guess which signer $i^\star$ will be $\mathcal{A}_4$'s prey beforehand. At the outset of the game, it thus chooses an index $i^\star \xleftarrow{R} \{1, \ldots, N\}$. Signer $i^\star$'s public key is set as $X_{i^\star} = A = g^a$. All other signers $i \neq i^\star$ are assigned public keys $X_i = g^{x_i}$ for which $\mathcal{B}_4$ knows the matching secret x_i and can thus always answer signing queries.
Hash queries and signing queries involving $i^\star$ are handled as in the proof of limited proxy security. When faced with a re-signing query from i to j for a valid signature $\sigma^{(\ell)}$ at level $\ell \in \{1, \ldots, L\}$, $\mathcal{B}_4$ ignores $\sigma^{(\ell)}$ and simulates a first level signature for signer j. The resulting signature $\sigma'^{(1)}$ is then turned into a $(\ell+1)^{\text{th}}$-level signature and given back to $\mathcal{A}_4$. A re-signing query thus triggers a signing query that only causes failure if $H(m)$ differs from g^μ for a known $\mu \in \mathbb{Z}_p^*$.
When $\mathcal{A}_4$ forges a signature at level L, $\mathcal{B}_4$ successfully extracts a $(2L-1)$-Flexible Diffie-Hellman tuple (as $\mathcal{B}_1$ and $\mathcal{B}_2$ did) if $H(m^\star) = (g^b)^{\mu^\star}$ and if it correctly guessed the identity $i^\star$ of the target signer. If $\mathcal{A}_4$'s advantage is ε, we find $O(\varepsilon/(N(q_s+q_{rs}+1)))$ as a lower bound on $\mathcal{B}_4$'s probability of success, q_s and q_{rs} being the number of signature and re-signature queries respectively. ∎

B.5 A Scheme in the Standard Model

Several extensions of BLS signatures have a standard model counterpart when Waters' technique supersedes random oracle manipulations (e.g. [LOS+06]). Likewise, we can very simply twist our method and achieve the first unidirectional PRS scheme (even including single hop ones) that avoids resorting to the random oracle model.

The scheme is, *mutatis mutandis*, quite similar to our first construction. Standard model security thus comes at the expense of a trusted setup to generate system parameters.

B.5.1 The Single Hop Variant

As in [Wat05], n denotes the length of messages to be signed. Arbitrary long messages can be signed if we first apply a collision-resistant hash function with n-bit outputs, in which case n is part of the security parameter.

The scheme requires a trusted party to generate common public parameters. However, this party can remain off-line after the setup phase.

Global-setup(λ, n)**:** given security parameters λ, n, this algorithm picks bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$, generators $g, h \xleftarrow{R} \mathbb{G}$ and a random $(n+1)$-vector $\overline{u} = (u', u_1, \ldots, u_n) \xleftarrow{R} \mathbb{G}^{n+1}$. The latter defines a function $F : \{0,1\}^n \to \mathbb{G}$ mapping n-bit strings $\mathsf{m} = m_1 \ldots m_n$ (where $m_i \in \{0,1\}$ for all $i \in \{0,1\}$) onto $F(\mathsf{m}) = u' \cdot \prod_{i=1}^{n} u_i^{m_i}$. The public parameters are

$$\mathsf{cp} := \{\mathbb{G}, \mathbb{G}_T, g, h, \overline{u}\}.$$

Keygen(λ)**:** user i sets his public key as $X_i = g^{x_i}$ for a random $x_i \xleftarrow{R} \mathbb{Z}_p^*$.

ReKeygen(x_j, X_i)**:** given user j's private key x_j and user i's public key X_i, generate the unidirectional re-signature key $R_{ij} = X_i^{1/x_j} = g^{x_i/x_j}$ that will be used to translate signature from i into signatures from j.

Sign$(1, \mathsf{m}, x_i)$**:** to sign a message $\mathsf{m} = m_1 \ldots m_n \in \{0,1\}^n$ at level 1, pick $r \xleftarrow{R} \mathbb{Z}_p^*$ at random and compute

$$\sigma^{(1)} = (\sigma_0, \sigma_1) = (h^{x_i} \cdot F(\mathsf{m})^r, g^r)$$

Sign$(2, \mathsf{m}, x_i)$**:** to generate a second level signature on $\mathsf{m} = m_1 \ldots m_n \in \{0,1\}^n$, choose $r, t \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3) = (h^{tx_i} \cdot F(\mathsf{m})^r, g^r, X_i^t, g^t)$$

Re-Sign$(1, \mathsf{m}, \sigma^{(1)}, R_{ij}, X_i, X_j)$**:** on input of a message $\mathsf{m} \in \{0,1\}^n$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a signature $\sigma^{(1)} = (\sigma_0, \sigma_1)$ and public keys X_i, X_j, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing if

$$e(\sigma_0, g) = e(X_i, h) \cdot e(F(\mathsf{m}), \sigma_1) \tag{B.3}$$

If $\sigma^{(1)}$ is a valid, it can be turned into a signature on behalf of j by choosing $r', t \xleftarrow{R} \mathbb{Z}_p^*$ and computing

$$\begin{aligned} \sigma^{(2)} &= (\sigma'_0, \sigma'_1, \sigma'_2, \sigma'_3) = (\sigma_0^t \cdot F(\mathsf{m})^{r'}, \sigma_1^t \cdot g^{r'}, X_i^t, R_{ij}^t) \\ &= (h^{tx_i} \cdot F(\mathsf{m})^{r''}, g^{r''}, X_i^t, g^{tx_i/x_j}) \end{aligned}$$

where $r'' = tr + r'$. If we set $\tilde{t} = tx_i/x_j$, we have

$$\sigma^{(2)} = (\sigma'_0, \sigma'_1, \sigma'_2, \sigma'_3) = (h^{\tilde{t}x_j} \cdot F(\mathsf{m})^{r''}, g^{r''}, X_j^{\tilde{t}}, g^{\tilde{t}}).$$

Verify$(1, \mathsf{m}, \sigma^{(1)}, X_i)$**:** the validity of a 1st level signature $\sigma^{(1)} = (\sigma_1, \sigma_2)$ is checked by testing if (B.3) holds.

Verify$(2, \mathsf{m}, \sigma^{(2)}, X_i)$**:** a signature $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3)$ at level 2 is accepted for the public key X_i if the following conditions are true.

$$\begin{aligned} e(\sigma_0, g) &= e(\sigma_2, h) \cdot e(F(\mathsf{m}), \sigma_1') \\ e(\sigma_2, g) &= e(X_i, \sigma_3). \end{aligned}$$

To the best of our knowledge, the above scheme is the first unidirectional PRS in the standard model and solves another problem left open in [AH05] where all constructions require the random oracle model. Like the scheme of section B.4, it can be scaled into a multi-hop system.

B.5.2 The Multi-Hop Extension

At levels $\ell \geq 2$, algorithms Sign, Re-Sign and Verify are generalized as follows.

Sign$(\ell + 1, m, x_i)$**:** to sign $\mathsf{m} \in \{0,1\}^n$ at level $\ell + 1$, user i picks $r \xleftarrow{R} \mathbb{Z}_p^*$, $(t_1, \ldots, t_\ell) \xleftarrow{R} (\mathbb{Z}_p^*)^\ell$ and outputs $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell+1}) \in \mathbb{G}^{2\ell+2}$ where

$$\begin{cases} \sigma_0 = h^{x_i t_1 \cdots t_\ell} \cdot F(\mathsf{m})^r & \\ \sigma_1 = g^r & \\ \sigma_k = g^{x_i t_1 \cdots t_{\ell+2-k}} & \text{for } k \in \{2, \ldots, \ell+1\} \\ \sigma_k = g^{t_{k-\ell-1}} & \text{for } k \in \{\ell+2, \ldots, 2\ell+1\}. \end{cases}$$

Re-Sign$(\ell + 1, \mathsf{m}, \sigma^{(\ell+1)}, R_{ij}, X_i, X_j)$**:** on input of a message $\mathsf{m} \in \{0,1\}^*$, the re-signature key $R_{ij} = g^{x_i/x_j}$, a purported $(\ell+1)^{\text{th}}$-level signature

$$\begin{aligned} \sigma^{(\ell+1)} &= (\sigma_0, \ldots, \sigma_{2\ell+1}) \\ &= (h^{x_i t_1 \cdots t_\ell} \cdot F(\mathsf{m})^r, g^r, g^{x_i t_1 \cdots t_\ell}, g^{x_i t_1 \cdots t_{\ell-1}}, \ldots, g^{x_i t_1}, g^{t_1}, \cdots, g^{t_\ell}) \in \mathbb{G}^{2\ell+2} \end{aligned}$$

and public keys X_i, X_j, check the correctness of $\sigma^{(\ell+1)}$ under X_i. If valid, $\sigma^{(\ell+1)}$ is translated for X_j by sampling $r' \xleftarrow{R} \mathbb{Z}_p^*$, $(r_0, r_1, \ldots, r_\ell) \xleftarrow{R} (\mathbb{Z}_p^*)^{\ell+1}$ and setting $\sigma^{(\ell+2)} = (\sigma_0', \ldots, \sigma_{2\ell+3}') \in \mathbb{G}^{2\ell+4}$ where

$$\begin{cases} \sigma_0' = \sigma_0^{r_0 \cdots r_\ell} \cdot F(\mathsf{m})^{r'} & \\ \sigma_1' = \sigma_1^{r_0 \cdots r_\ell} \cdot g^{r'} & \\ \sigma_k' = \sigma_k^{r_0 \cdots r_{\ell+2-k}} & \text{for } k \in \{2, \ldots, \ell+1\} \\ \sigma_{\ell+2}' = X_i^{r_0} & \\ \sigma_{\ell+3}' = R_{ij}^{r_0} & \\ \sigma_k' = \sigma_{k-2}^{r_{k-\ell-3}} & \text{for } k \in \{\ell+4, \ldots, 2\ell+3\}. \end{cases}$$

If we define $\tilde{t}_0 = r_0 x_i / x_j$, $r'' = r_0 \cdots r_\ell + r'$ and $\tilde{t}_k = r_k t_k$ for $k = 1, \ldots, \ell$, we observe that

$$\sigma^{(\ell+2)} = (h^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell} \cdot F(\mathsf{m})^{r''}, g^{r''}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_\ell}, g^{x_j \tilde{t}_0 \tilde{t}_1 \cdots \tilde{t}_{\ell-1}}, \ldots, g^{x_j \tilde{t}_0}, g^{\tilde{t}_0}, \ldots, g^{\tilde{t}_\ell}).$$

Verify$(\ell + 1, \mathsf{m}, \sigma^{(\ell+1)}, X_i)$**:** a candidate signature $\sigma^{(\ell+1)} = (\sigma_0, \ldots, \sigma_{2\ell+1})$ is verified by testing if the following equalities hold:

$$\begin{aligned} e(\sigma_0, g) &= e(h, \sigma_3) \cdot e(F(\mathsf{m}), \sigma_1) \\ e(\sigma_k, g) &= e(\sigma_{k+1}, \sigma_{2\ell+3-k}) \text{ for } k \in \{2, \ldots, \ell\} \\ e(\sigma_{\ell+1}, g) &= e(X_i, \sigma_{\ell+2}). \end{aligned}$$

B.5.3 Security

Theorem B.5.1 The scheme with L levels (and thus at most $L-1$ hops) is a secure unidirectional PRS under the $(L-1)$-FlexDH and mCDH assumptions.

Proof: The proof is very similar to the one of theorem B.4.1. ▮

B.6 Single-Hop Schemes in the Chosen Key Model

This section shows a simple way to modify the single-hop versions of our schemes so as to prove their security in the plain public key model and dispense with the knowledge of secret key assumption. We outline the required modifications in our first scheme but they can be applied to our standard model system as well.

The idea is to randomize the generation of re-signature keys, the shape of which becomes reminiscent of Waters signatures. Using techniques that were initially proposed for identity-based encryption [BB04a], we can then prove security results without positioning ourselves in the KOSK model.

Global-setup(λ)**:** is as in section B.4.

Keygen(λ)**:** user i's public key is $pk_i = (X_i = g^{x_i}, Y_i = g^{y_i})$ for random $x_i, y_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$.

ReKeygen(x_j, y_j, pk_i)**:** given x_j, y_j and $pk_i = (X_i, Y_i)$, this algorithm outputs the re-signature key

$$R_{ij} = (R_{ij1}, R_{ij2}) = (X_i^{1/x_j} \cdot Y_j^r,\ X_j^r)$$

for a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and where $(X_j, Y_j) = (g^{x_j}, g^{y_j})$.

Sign$(1, x_i, m)$**:** outputs $\sigma^{(1)} = H(m)^{x_i} \in \mathbb{G}$ as in section B.4.

Sign$(2, x_i, m)$**:** to sign $m \in \{0,1\}^*$ at level 2, user i chooses $s, t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\begin{aligned} \sigma^{(2)} &= (\sigma_0, \sigma_1, \sigma_2, \sigma_3) \\ &= (H(m)^{x_i t},\ X_i^t,\ g^t \cdot Y_i^s,\ X_i^s). \end{aligned}$$

Re-Sign$(1, m, \sigma^{(1)}, R_{ij}, pk_i, pk_j)$**:** given the re-signature key $R_{ij} = (R_{ij1}, R_{ij2})$, a signature $\sigma^{(1)} \in \mathbb{G}$ and public keys $pk_i = (X_i, Y_i)$, $pk_j = (X_j, Y_j)$, check the validity of $\sigma^{(1)}$ w.r.t signer i by testing $e(\sigma^{(1)}, g) = e(H(m), X_i)$. If valid, $\sigma^{(1)}$ is turned into a signature on behalf of j by choosing $s', t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computing

$$\begin{aligned} \sigma^{(2)} &= (\sigma'_0, \sigma'_1, \sigma'_2, \sigma'_3) \\ &= ({\sigma^{(1)}}^t,\ X_i^t,\ R_{ij1}^t \cdot Y_j^{s'},\ R_{ij2}^t \cdot X_j^{s'}) \\ &= (H(m)^{x_i t},\ X_i^t,\ g^{t x_i / x_j} \cdot Y_j^{rt+s'},\ X_j^{rt+s'}) \end{aligned}$$

If we set $\tilde{t} = t x_i / x_j$ and $\tilde{s} = rt + s'$, we have

$$\sigma^{(2)} = (H(m)^{x_j \tilde{t}},\ X_j^{\tilde{t}},\ g^{\tilde{t}} \cdot Y_j^{\tilde{s}},\ X_j^{\tilde{s}}).$$

Verify$(1, m, \sigma^{(1)}, pk_i)$**:** accept $\sigma^{(1)}$ if $e(\sigma^{(1)}, g) = e(H(m), X_i)$.

Verify$(2, m, \sigma^{(2)}, pk_i)$**:** accept $\sigma^{(2)} = (\sigma_0, \sigma_1, \sigma_2, \sigma_3)$ w.r.t. $pk_i = (X_i, Y_i)$ if the following relations hold.

$$e(\sigma_0, g) = e(\sigma_1, H(m)) \quad e(\sigma_2, X_i) = e(g, \sigma_1) \cdot e(Y_i, \sigma_3)$$

The above scheme features a comparable efficiency to the one of section B.4 with signatures that are only slightly longer at level 2. We were unfortunately unable to turn it into a multi-hop system.

From a security standpoint, we also need fewer assumptions in the proofs since the 1-Flexible Diffie-Hellman assumption suffices.

Theorem B.6.1 The single-hop scheme is secure in the chosen-key model under the 1-FlexDH assumption.

Proof: We can prove the result without resorting to the modified CDH assumption and using only the 1-Flexible Diffie-Hellman problem. Let $(g, A = g^a, B = g^b)$ be a given instance of the latter.

External security and limited proxy security For these notions, the proofs work out almost exactly as in the proof of theorem B.4.1. The only difference is in the generation of users' public keys $pk_i = (X_i, Y_i)$: the first component X_i is chosen as in the proof of theorem B.4.1 whilst Y_i is set as $Y_i = X_i^{y_i}$ for randomly drawn exponents $y_i \xleftarrow{R} \mathbb{Z}_p^*$. When the adversary eventually outputs a forgery

$$(\sigma_0^\star, \sigma_1^\star, \sigma_2^\star, \sigma_3^\star) = (H(m^\star)^{axt},\ X^t,\ g^t \cdot Y^r,\ X^r),$$

w.r.t. to a honest user's public key $(X = A^x, Y = X^y)$ (where $x, y \in \mathbb{Z}_p^*$ are random exponents initially chosen by the simulator), one can compute $(\sigma_0^\star, \sigma_1^\star, \sigma_2^\star/\sigma_3^{\star y})$ and use it as a forgery against our scheme of section B.4. Namely, if $H(m^\star) = B^{\mu^\star}$ and $X = A^x$ for known values $x, \mu^\star \in \mathbb{Z}_p^*$, the simulator obtains a triple

$$(C^{ab}, C^a, C) = \Big(\sigma_0^\star,\ \sigma_1^{\star\mu^\star},\ (\frac{\sigma_2^\star}{\sigma_3^{\star y}})^{x\mu^\star}\Big),$$

which solves the problem instance.

Delegatee security The proof is as in theorem B.4.1 but the adversary is given a single honest user's public key.

Delegator security From an adversary $\mathcal{A}$ with advantage ε and making q_s signing queries, we build an algorithm $\mathcal{B}$ that finds g^{ab} with probability $O(\varepsilon/q_s)$.

System parameters: $\mathcal{A}$ is provided with public parameters $\{\mathbb{G}, \mathbb{G}_T, g, \mathcal{O}_H\}$ where $\mathcal{O}_H$ is the random oracle.

Key generation: the delegator's public key is defined as $pk_0 = (X_0, Y_0) = (A, g^{y_0})$ for a random $y_0 \xleftarrow{R} \mathbb{Z}_p^*$.

Oracle queries:

- $\mathcal{A}$'s random oracle queries and signing queries are handled using Coron's technique [Cor00] as in the proof of theorem B.4.1 (we have thus again a degradation factor of $O(q_s)$ in the reduction).
- *Delegation queries*: at any time $\mathcal{A}$ can supply a public key $pk = (X, Y)$ (without having to reveal the underlying secret) and ask oracle $\mathcal{O}_{dlg}(.)$ to generate a re-signature key on behalf of the delegator 0 using pk as the delegatee's public key. Since we have $(X_0 = A, Y_0 = g^{y_0})$ for a known exponent y_0, $\mathcal{B}$ picks $r \xleftarrow{R} \mathbb{Z}_p^*$ and returns

$$(R_1, R_2) = (g^{ry_0},\ X_0^r \cdot X^{-1/y_0}). \tag{B.4}$$

  If we define $\tilde{r} = r - x/(ay_0)$, where $x = \log_g(X)$, we see that (R_1, R_2) has the correct shape since

$$X^{1/a} \cdot Y_0^{\tilde{r}} = X^{1/a} \cdot Y_0^r \cdot (g^{y_0})^{-\frac{x}{ay_0}} = g^{ry_0}$$

  and $X_0^{\tilde{r}} = X_0^r \cdot A^{-\frac{x}{ay_0}} = X_0^r \cdot X^{-1/y_0}$. We observe that $\mathcal{B}$ can compute both parts of (B.4) without knowing $x = \log_g(X)$ or $y = \log_g(Y)$.

After a number of queries, $\mathcal{A}$ comes up with a first level forgery that allows computing g^{ab} as in the proof of theorem B.4.1. Unlike what happens in the latter, $\mathcal{B}$ does not need $g^{1/a}$ at any time during the simulation and we only need the 1-Flexible Diffie-Hellman assumption. ▮

B.7 Can one achieve constant-size multi-hop signatures?

While highly desirable, unidirectional multi-hop PRS with constant-size signatures turn out to be very hard to construct. We give arguments explaining why they seem out of reach with the current state of knowledge.

Trivially, if the Re-Sign algorithm increases the size of signatures (even by a single bit), then we inevitably end up with a linear size in the number of delegations. Intuitively, multi-hop unidirectional systems therefore provide either constant or linear sizes. It seems very unlikely that one will be able to come up with logarithmic-size signatures for instance. This apparently indicates that, regardless of how many times signatures get translated, they should remain in the same signature space (which sounds hardly compatible with the pursued unidirectionality). Nonetheless, not all unidirectional schemes do lengthen signatures upon translation: if implemented with appropriate parameters, the first proposal of [AH05] features the same signature size at both levels (though signatures have different shapes). However, it does not lend itself to a multi-use extension: to translate a signature, the proxy uses a piece of it as an exponent to exponentiate the re-signature key, which hampers length-preserving re-iterations of the process.

Up to now, all known unidirectional proxy re-cryptography primitives make use of bilinear maps. Unfortunately, those tools still fall short of reaching the aforementioned purpose. Pairing-based schemes often let proxies replace a component of the original ciphertext or signature by its pairing with the proxy key. Multiple hops are impossible if we leave the resulting pairing value inside the re-signature since no bilinear map is defined over the target group $\mathbb{G}_T$. To circumvent this issue, our approach postpones the computation of the pairing until the verification by blinding its arguments and introducing them into transformed signatures. Unfortunately, this inevitably increases their length at each conversion.

We are not claiming that constant signature sizes are impossible to obtain. But it turns out that new ideas and techniques should be developed to reach this goal.

B.8 Generic hardness of ℓ-FlexDH in bilinear groups

To provide more confidence in the ℓ-FlexDH assumption we give a lower bound on the computational complexity of the ℓ-FlexDH problem for generic groups equipped with bilinear maps. In [KJP06], Kunz-Jacques and Pointcheval define a family of computational problems that enables to study variants of the CDH problem in the generic group model. Let $\mathcal{A}$ be an adversary in this model and $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$ be a multivariate polynomial whose coefficients might depend on $\mathcal{A}$'s behavior. For values of $x_1, \ldots, x_k$ chosen by the simulator, and knowing their encodings, the goal of $\mathcal{A}$ is to compute the encodings of $y_1, \ldots, y_\ell$ such that

$$\varphi(x_1, \ldots, x_k, y_1, \ldots, y_\ell) = 0.$$

All elements manipulated by $\mathcal{A}$ are linear polynomials in $x_1, \ldots, x_k$ and some new random elements introduced via the group oracle. Let us denote P_i the polynomial corresponding to y_i (it is a random variable), Kunz-Jacques and Pointcheval proved the following result.

Theorem B.8.1 [[KJP06]] Let $d = \deg(\varphi)$ and $\mathsf{P_m}$ be an upper bound for the probability

$$\Pr[\varphi(X_1, \ldots, X_k, P_1(X_1, \ldots, X_k), \ldots, P_\ell(X_1, \ldots, X_k)) = 0]$$

Then the probability that $\mathcal{A}$ wins after q_G queries satisfies

$$\mathsf{Succ}(q_G) \leq \mathsf{P_m} + \frac{(3q_G + k + 2)}{2p} + \frac{d}{p}.$$

The choice $\phi(X_1, X_2, Y_1, \ldots, Y_{\ell+1}) = Y_{\ell+1} - X_1 X_2 Y_1 \ldots Y_\ell$ implies the generic hardness of the ℓ-FlexDH problem. It is almost straightforward to prove that the Kunz-Jacques-Pointcheval result also holds in generic bilinear groups where the ℓ-FlexDH problem thus remains intractable. The details are given in the full version of the paper.

Theorem B.8.2 Let $d = \deg(\varphi)$ and P_m be an upper bound for the probability

$$\Pr[\varphi(X_1, \ldots, X_k, P_1(X_1, \ldots, X_k), \ldots, P_\ell(X_1, \ldots, X_k)) = 0]$$

Then the probability that $\mathcal{A}$ wins after q_G oracle queries to the group operations in $\mathbb{G}$, $\mathbb{G}_T$ to the bilinear map e satisfies

$$\mathsf{Succ}(q_G) \leq P_m + \frac{(3q_G + k + 2)}{p} + \frac{d}{p}.$$

B.9 Conclusions and Open Problems

We described the first multi-use unidirectional proxy re-signatures, which solves a problem left open at CCS 2005. Our random-oracle-based proposal also offers efficiency improvements over existing solutions at the first level. The other scheme additionally happens to be the first unidirectional PRS in the standard model. We finally showed how to construct single-hop schemes in the chosen-key model.

Two major open problems remain. First, it would be interesting to see if multi-level unidirectional PRS have efficient realizations under more classical intractability assumptions. A perhaps more challenging task would be to find out implementations – if they exist at all – of such primitives where the size of signatures and the verification cost do not grow linearly with the number of translations.

Acknowledgements

The authors thank Mark Manulis and the anonymous referees for their comments. The first author acknowledges the support of the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.). The second author is supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by the French *Agence Nationale de la Recherche* through the PACE project.

Appendix C

Lossy Encryption: Constructions from General Assumptions and Efficient Selective Opening Chosen Ciphertext Security

Asiacrypt 2011
[HLOV11] with B. Hemenway, B. Libert and R. Ostrovsky

Abstract : Lossy encryption was originally studied as a means of achieving efficient and composable oblivious transfer. Bellare, Hofheinz and Yilek showed that lossy encryption is also selective opening secure. We present new and general constructions of lossy encryption schemes and of cryptosystems secure against selective opening adversaries.

We show that *every* re-randomizable encryption scheme gives rise to efficient encryptions secure against a selective opening adversary. We show that statistically-hiding 2-round Oblivious Transfer implies Lossy Encryption and so do smooth hash proof systems. This shows that private information retrieval and homomorphic encryption *both* imply Lossy Encryption, and thus Selective Opening Secure Public Key Encryption.

Applying our constructions to well-known cryptosystems, we obtain selective opening secure commitments and encryptions from the *Decisional Diffie-Hellman*, *Decisional Composite Residuosity* and *Quadratic Residuosity* assumptions.

In an indistinguishability-based model of chosen-ciphertext selective opening security, we obtain secure schemes featuring short ciphertexts under standard number theoretic assumptions. In a simulation-based definition of chosen-ciphertext selective opening security, we also handle non-adaptive adversaries by adapting the Naor-Yung paradigm and using the perfect zero-knowledge proofs of Groth, Ostrovsky and Sahai.

C.1 Introduction

In Byzantine agreement, and more generally in secure multiparty computation, it is often assumed that all parties are connected to each other via private channels. In practice, these private channels are implemented using a public-key cryptosystem. An adaptive adversary in a MPC

setting, however, has very different powers than an adversary in an IND-CPA or IND-CCA game. In particular, an adaptive MPC adversary may view all the encryptions sent in a given round, and then choose to corrupt a certain fraction of the players, thus revealing the decryptions of those players' messages *and the randomness used to encrypt them.* A natural question is whether the messages sent from the uncorrupted players remain secure. If the messages (and randomness) of all the players are chosen independently, then security in this setting follows immediately from the IND-CPA security of the underlying encryption. If, however, the messages are not chosen independently, the security does not immediately follow from the IND-CPA (or even IND-CCA) security of the underlying scheme. In fact, although this problem was first investigated over twenty years ago, it remains an open question whether IND-CPA (or IND-CCA) security implies this *selective opening* security.

A similar question may be asked regarded in terms of commitments as well. Suppose an adversary is allowed to see commitments to a number of related messages, the adversary may then choose a subset of the commitments for the challenger to de-commit. Does this reveal any information about the unopened commitments? This question has applications to concurrent zero-knowledge proofs.

C.1.1 Related Work

Prior results. There have been many attempts to design encryption protocols that can be used to implement secure multiparty computation against an adaptive adversary. The first protocols by Beaver and Haber [BH92] required interaction between the sender and receiver, required erasure and were fairly inefficient. The first non-interactive protocol was given by Canetti, Feige, Goldreich and Naor in [CFGN96]. In [CFGN96] the authors defined a new primitive called Non-Committing Encryption, and gave an example of such a scheme based on the RSA assumption. In [Bea97], Beaver extended the work of [CFGN96], and created adaptively secure key exchange under the Diffie-Hellman assumption. In subsequent work, Damgård and Nielsen improved the efficiency of the schemes of Canetti *et al.* and Beaver, they were also able to obtain Non-Committing Encryption based on one-way trapdoor functions with invertible sampling. In [CHK05a], Canetti, Halevi and Katz presented a Non-Committing encryption protocols with evolving keys.

In [CDNO97], Canetti, Dwork, Naor and Ostrovsky extended the notion of Non-Committing Encryption to a new protocol which they called Deniable Encryption. In Non-Committing Encryption schemes there is a simulator, which can generate non-committing ciphertexts, and later open them to any desired message, while in Deniable Encryption, valid encryptions generated by the sender and receiver can later be opened to any desired message. The power of this primitive made it relatively difficult to realize, and Canetti *et al.* were only able to obtain modest examples of Deniable Encryption and left it as an open question whether fully deniable schemes could be created.

The notions of security against an adaptive adversary can also be applied to commitments. In fact, according to [DNRS03] the necessity of adaptively-secure commitments was realized by 1985. Despite its utility, until recently, relatively few papers directly addressed the question of commitments secure against a selective opening adversary (SOA). The work of Dwork, Naor, Reingold and Stockmeyer [DNRS03] was the first to explicitly address the problem. In [DNRS03], Dwork *et al.* showed that non-interactive SOA-secure commitments can be used to create a 3-round zero-knowledge proof systems for NP with negligible soundness error, and they gave constructions of a weak form of SOA-secure commitments, but leave open the question of whether general SOA-secure commitments exist.

The question of SOA-secure commitments was put on firm foundations by Hofheinz [Hof11] and Bellare, Hofheinz and Yilek in [BHY09]. In [BHY09], Bellare *et al.* distinguished between

simulation-based and indistinguishability-based definitions of security, and gave a number of constructions and black-box separations. In particular, Hofheinz showed that, in the simulation-based setting, non-interactive SOA-secure commitments cannot be realized in a black-box manner from standard cryptographic assumptions, but if interaction is allowed, they can be created from one-way permutations in a non-black-box manner. In the indistinguishability-based setting, they showed that any statistically-hiding scheme achieves this level of security, but that there is a black-box separation between perfectly-binding SOA-secure commitments and most standard cryptographic assumptions. Our results in the selective opening setting build on the breakthrough results of [BHY09].

Independent and concurrent work. Fehr, Hofheinz and Kiltz and Wee [FHKW10] also investigate the case of CCA2 cryptosystems that are selective opening secure. In their work, they show how to adapt the universal hash proof systems of [CS02], to provide CCA2 security in the selective opening setting. Their constructions are general, and offer the first SEM-SO-CCA secure cryptosystem whose parameters are completely independent of n, the number of messages. Fehr *et al.* [FHKW10] also consider selective opening security against chosen-plaintext attacks, and using techniques from Non-Committing Encryption [CFGN96] they construct SEM-SO-CPA secure systems from enhanced one-way trapdoor permutations.

The results of Bellare, Waters and Yilek [BWY11] show how to construct Identity-Based Encryption (IBE) schemes secure under selective-opening attacks based on the Decision Linear Assumption. Our work is orthogonal to theirs. Their work constructs IBE schemes secure under selective-opening attacks, while our work starts with a tag-based encryption scheme, and uses it to construct encryption schemes that are secure against a selective-opening chosen-ciphertext attack, but are not identity-based.

Recent results. While this paper was being accepted at Asiacrypt 2011, Bellare, Dowsley, Waters and Yilek [BDWY12] provided separation results between semantic security and selective opening security. They demonstrated the existence of semantically secure public-key encryption schemes that *cannot* be proved SOA-secure in the sense of a simulation-based definition. More precisely, assuming the availability of collision-resistant hash functions, Bellare *et al.* [BDWY12] showed that, if a commitment or encryption scheme is committing[1] and non-interactive, there exists a selective opening adversary for which no simulator – let alone black-box – can meet the security definition. In the case of indistinguishability-based definitions, it remains an open question whether a similar separation can be found for restricted message distributions.

In further steps towards a better understanding of SOA security, Böhl, Hofheinz and Kraschewski [BHK12] showed that simulation-based definitions and enhanced indistinguishability-based definitions are incomparable. Namely, [BHK12] describes an encryption scheme that is provably simulation-based SOA-secure but fails to satisfy a notion called full indistinguishability-based SOA (or full IND-SO-CPA for short) security. Here, "full" refers to the fact that the entity running the IND-SO-CPA experiment does not have to be efficient. In the converse direction [BHK12] proved that, if a fully IND-SOA secure encryption scheme exists at all, it can be turned into another encryption scheme for which the results of [BDWY12] rule out the existence of a simulator establishing simulation-based SOA security.

[1] In the context of encryption, "committing" means that, even if a computationally bounded adversary generates the public key itself, he cannot find two message-randomness pairs giving the same ciphertext. The Elgamal cryptosystem is a simple example of binding encryption scheme.

C.1.2 Our Contributions

Lossy Encryption under General Assumptions. In this paper, we primarily consider encryptions secure against a selective opening adversary. In particular, we formalize the notion of re-randomizable Public-Key Encryption and we show that re-randomizable encryption implies Lossy Encryption, as defined in [PVW08] and expanded in [BHY09]. Combining this with the recent result of Bellare, Hofheinz and Yilek [BHY09] showing that Lossy Encryption is IND-SO-ENC secure, we have an efficient construction of IND-SO-ENC secure encryption from any re-randomizable encryption (which generalizes and extends previous results). Furthermore, these constructions retain the efficiency of the underlying re-randomizable encryption protocol.

Applying our results to the Paillier cryptosystem [Pai99], we obtain an encryption scheme which attains a strong, simulation-based form of semantic security under selective openings (SEM-SO-ENC security). This is the first construction of this type from the Decisional Composite Residuosity (DCR) assumption. As far as bandwidth goes, it is also the most efficient SEM-SO-ENC secure encryption scheme to date. We note that the possible use of Paillier as a lossy encryption scheme was implicitly mentioned in [YY05]. To the best of our knowledge, its SEM-SO-ENC security was not reported earlier.

We go on to show that Lossy Encryption is also implied by (honest-receiver) statistically-hiding $\binom{2}{1}$-Oblivious Transfer and by hash proof systems [CS02]. Combining this with the results of [PVW08], we recognize that Lossy Encryption is essentially just a different way to view the well known statistically-hiding $\binom{2}{1}$-OT primitive. Applying the reductions in [BHY09] to this result, yields constructions of SOA secure encryption from both PIR and homomorphic encryption.

These results show that the Lossy and Selective Opening Secure Encryption primitives (at least according to the latter's indistinguishability-based security definition), which have not been extensively studied until recently, are actually implied by several well-known primitives: *i.e.*, re-randomizable encryption, PIR, homomorphic encryption, hash proof systems and statistically-hiding $\binom{2}{1}$-OT. Prior to this work, the only known general[2] constructions of lossy encryption were from lossy trapdoor functions. Our results thus show that they can be obtained from many seemingly weaker primitives (see figure C.1).

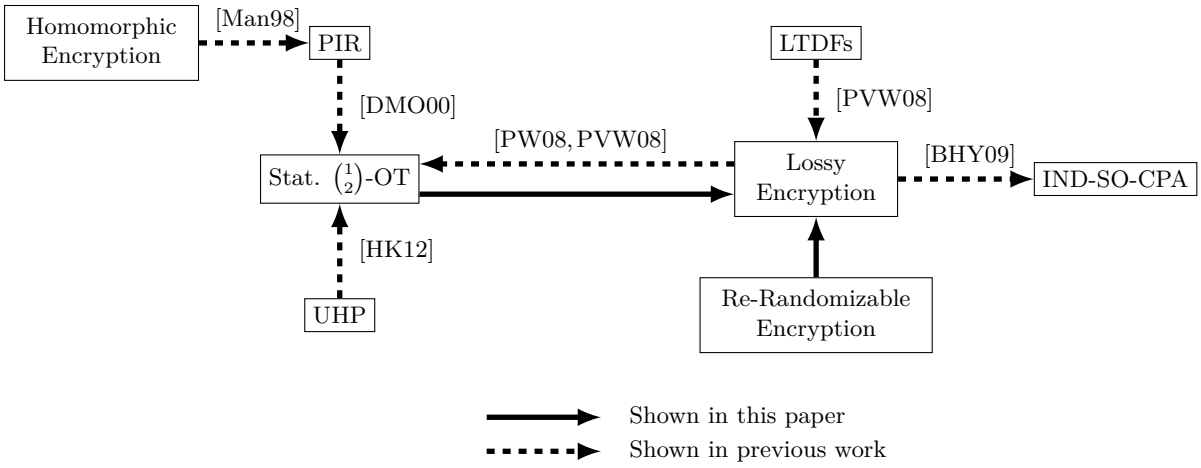


Figure C.1: Constructing Lossy Encryption

[2] *i.e.*, not based on specific number-theoretic assumptions

Selective Opening Security Against Chosen-Ciphertext Attacks. Continuing the study of selective-opening security, we present definitions chosen-ciphertext security (CCA2) in the selective opening setting (in both the indistinguishability and simulation-based models) and describe encryption schemes that provably satisfy these enhanced forms of security. Despite recent progress, relatively few methods are known for constructing IND-CCA2 cryptosystems in the standard model. The problem is even more complex with selective openings, where some known approaches for CCA2 security do not seem to apply. We note how the Naor-Yung paradigm, even when applied with statistical zero knowledge proofs fails to prove CCA2 security in the selective opening setting. Essentially, this is because the selective opening adversary learns the randomness used in the signature scheme, which allows him to forge signatures, and thus create ciphertexts that cannot be handled by the simulated decryption oracle.

The results of Fehr, Hofheinz, Kiltz and Wee [FHKW10] show how to modify universal hash proof systems [CS02] to achieve security under selective openings.

We take a different approach and follow (a variant of) the Canetti-Halevi-Katz paradigm [CHK04]. This too encounters many obstacles in the selective opening setting. Nevertheless, under standard assumptions (such as DDH or the Composite Residuosity assumption), we construct schemes featuring compact ciphertexts while resisting adaptive (*i.e.*, CCA2) chosen-ciphertext attacks according to our indistinguishability-based definition. When comparing our schemes to those of [FHKW10], we note that our public key size depends on n, the number of senders that can be possibly corrupted, while the systems of [FHKW10] are independent of n. On the other hand, to encrypt m-bit messages with security parameter λ, our ciphertexts are of length $\mathcal{O}(\lambda + m)$, while theirs are of length $\mathcal{O}(\lambda m)$. Our public-keys are longer than in [FHKW10] because our construction relies on All-But-N Lossy Trapdoor Functions (defined below), which have long description. The recent complementary work of Hofheinz [Hof12] shows how to create All-But-Many Trapdoor Functions with short keys. Using his results in our construction eliminates the dependence of the public-key size on n. Regarding security definitions, our constructions satisfy an indistinguishability-based definition (IND-SO-CCA), whereas theirs fit a simulation-based definition (SEM-SO-CCA) which avoids the restriction on the efficient conditional re-sampleability of the message distribution.

The scheme of [FHKW10] is very different from ours and we found it interesting to investigate the extent to which well-known paradigms like [CHK04] can be applied in the present context. Moreover, by adapting the Naor-Yung paradigm [NY90], under more general assumptions, we give a CCA1 construction that also satisfies a strong simulation-based notion of adaptive selective opening security.

One advantage of our IND-SO-CCA scheme is the ability to natively encrypt multi-bit messages. It is natural to consider whether our approach applies to the scheme of Bellare, Waters and Yilek [BWY11] to achieve multi-bit IND-SO-CCA encryption. The scheme of [BWY11], like [FHKW10], encrypts multi-bit messages in a bitwise manner. Applying a Canetti-Halevi-Katz-like transformation to the construction of [BWY11] does not immediately yield IND-SO-CCA encryption schemes for multi-bit messages: the reason is that it is not clear how to prevent the adversary from reordering the bit encryptions without employing a one-time signature scheme.

C.2 Background

C.2.1 Notation

If $f : X \to Y$ is a function, for any $Z \subset X$, we let $f(Z) = \{f(x) : x \in Z\}$. If A is a PPT machine, then we use $a \xleftarrow{\$} A$ to denote running the machine A and obtaining an output, where a

is distributed according to the internal randomness of A. For a PPT machine A, we use $\mathsf{coins}(A)$ to denote the distribution of the internal randomness of A. So the distributions $\{a \xleftarrow{\$} A\}$ and $\{r \xleftarrow{\$} \mathsf{coins}(A) : a = A(r)\}$ are identical. If R is a set, we use $r \xleftarrow{\$} R$ to denote sampling uniformly from R.

If X and Y are families of distributions indexed by a security parameter λ, we use $X \approx_s Y$ to mean the distributions X and Y are statistically close, *i.e.*, for all polynomials p and sufficiently large λ, we have $\sum_x |\Pr[X = x] - \Pr[Y = x]| < \frac{1}{p(\lambda)}$.

We use $X \approx_c Y$ to mean X and Y are computationally close, *i.e.*, for all PPT adversaries A, for all polynomials p, then for all sufficiently large λ, we have $|\Pr[A^X = 1] - \Pr[A^Y = 1]| < 1/p(\lambda)$.

C.2.2 Selective Opening Secure Encryption

We recall an indistinguishability-based definition of encryption secure against a selective opening adversary that was originally formalized in [BHY09]. We define two games, a real and an ideal game which should be indistinguishable to any efficient adversary. The key point to notice is that the adversary receives *both* the messages and the randomness for his selection. This mirrors the fact that an adaptive MPC adversary learns the entire history of corrupted players (*i.e.*, there are no secure erasures). If the adversary receives only the messages this would reduce to standard CPA security.

As in [BHY09], $\mathcal{M}$ denotes an n-message sampler outputting a n-vector $\mathbf{m} = (m_1, \ldots, m_n)$ of messages whereas $\mathcal{M}_{|I,\mathbf{m}[I]}$ denotes an algorithm that conditionally resamples another random n-vector $\mathbf{m}' = (m'_1, \ldots, m'_n)$ such that $m'_i = m_i$ for each $i \in I \subset \{1, \ldots, n\}$. If such a resampling can be done efficiently for all $I, \mathbf{m}$, then $\mathcal{M}$ is said to support efficient conditional resampling.

Definition C.2.1 (Indistinguishability under selective openings). A public key cryptosystem (G, E, D) is indistinguishable under selective openings (IND-SO-ENC secure) if, for any message sampler $\mathcal{M}$ supporting efficient conditional resampling and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have

$$\left|\Pr\left[\mathcal{A}^{\mathsf{ind\text{-}so\text{-}real}} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{ind\text{-}so\text{-}ideal}} = 1\right]\right| < \nu$$

for some negligible function ν, and where the games $\mathsf{ind\text{-}so\text{-}real}$ and $\mathsf{ind\text{-}so\text{-}ideal}$ are defined as follows.

IND-SO-ENC (Real)	**IND-SO-ENC (Ideal)**
$\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$	$\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$
$r_1, \ldots, r_n \xleftarrow{\$} \mathsf{coins}(E)$	$r_1, \ldots, r_n \xleftarrow{\$} \mathsf{coins}(E)$
(I, st)	(I, st)
$\quad \xleftarrow{\$} \mathcal{A}_1\big(pk, E(m_1, r_i), \ldots, E(m_n, r_n)\big)$	$\quad \xleftarrow{\$} \mathcal{A}_1\big(pk, E(m_1, r_i), \ldots, E(m_n, r_n)\big)$
$b \xleftarrow{\$} \mathcal{A}_2\big(st, (m_i, r_i)_{i \in I}, \mathbf{m}\big)$	$\mathbf{m}' = (m'_1, \ldots, m'_n) \xleftarrow{\$} \mathcal{M}_{\|I,\mathbf{m}[I]}$
	$b \xleftarrow{\$} \mathcal{A}_2\big(st, (m_i, r_i)_{i \in I}, \mathbf{m}'\big)$

Figure C.2: IND-SO-ENC security

In the real game, the challenger samples $\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$ from the joint message distribution. Then, it generates randomness $r_1, \ldots, r_n \xleftarrow{\$} \mathsf{coins}(E)$ and sends the vector of ciphertexts $(E(m_1, r_1), \ldots, E(m_n, r_n)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ responds with a subset $I \subset \{1, \ldots, n\}$ of size $\#I = n/2$. The challenger reveals r_i for each $i \in I$ as well as the *entire* vector $\mathbf{m} = (m_1, \ldots, m_n)$ to $\mathcal{A}$. Finally, the latter outputs a bit $b \in \{0, 1\}$.

In the ideal game, the challenger also samples $\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$ from the joint distribution. Then, it generates random coins $r_1, \ldots, r_n \xleftarrow{\$} \mathsf{coins}(E)$ and sends the vector of ciphertexts $(E(m_1, r_1), \ldots, E(m_n, r_n))$ to the adversary $\mathcal{A}$. The latter chooses a subset $I \subset \{1, \ldots, n\}$ with $\#I = n/2$ and the challenger reveals r_i for $i \in I$. The only difference w.r.t. the real game is that, instead of revealing $\mathbf{m}$, the challenger samples a new vector $\mathbf{m}' \xleftarrow{\$} \mathcal{M}_{|I,\mathbf{m}[I]}$ and sends $\mathbf{m}'$ to $\mathcal{A}$. Eventually, the adversary outputs a bit $b \in \{0, 1\}$.

We stress that the challenger reveals both the plaintexts m_i and the randomness r_i for indices $i \in I$. If only the messages m_i were revealed, this security would follow immediately from IND-CPA security.

C.2.3 Lossy Encryption

In [PVW08], Peikert, Vaikuntanathan and Waters defined Dual-Mode Encryption, a type of cryptosystem with two types public-keys, injective keys on which the cryptosystem behaves normally and "lossy" or "messy" keys on which the system loses information about the plaintext. In particular they require that the encryptions of any two plaintexts under a lossy key yield distributions that are statistically close, yet injective and lossy keys remain computationally indistinguishable.

In [BHY09] Bellare, Hofheinz and Yilek define *Lossy Encryption*, expanding on the definitions of Dual-Mode Encryption in [PVW08], and Meaningful/Meaningless Encryption in [KN08]. At a high level, a 'lossy' (or 'messy' in the terminology of [PVW08]) cryptosystem is one which has two types of public keys which specify two different modes of operation. In the normal mode, encryption is injective, while in the lossy (or 'messy') mode, the ciphertexts generated by the encryption algorithm are independent of the plaintext. We also require that no efficient adversary can distinguish normal keys from lossy keys. In [BHY09], they also require a property called *openability*, which basically allows a possibly inefficient algorithm to open a ciphertext generated under a lossy key to *any* plaintext.

Definition C.2.2 A *lossy public-key encryption scheme* is a tuple (G, E, D) of efficient algorithms such that

- $G(1^\lambda, \mathsf{inj})$ outputs keys (pk, sk), keys generated by $G(1^\lambda, \mathsf{inj})$ are called *injective keys*.
- $G(1^\lambda, \mathsf{lossy})$ outputs keys $(pk_{\mathsf{lossy}}, sk_{\mathsf{lossy}})$, keys generated by $G(1^\lambda, \mathsf{lossy})$ are called *lossy keys*.

Additionally, the algorithms must satisfy the following properties:

1. *Correctness on injective keys.* For all plaintexts $x \in X$,
$$\Pr\left[(pk, sk) \xleftarrow{\$} G(1^\lambda, \mathsf{inj});\ r \xleftarrow{\$} \mathsf{coins}(E) : D(sk, E(pk, x, r)) = x\right] = 1.$$
2. *Indistinguishability of keys.* In lossy mode, public keys are computationally indistinguishable from those in the injective mode. Specifically, if $\mathrm{proj} : (pk, sk) \mapsto pk$ is the projection map, then
$$\{\mathrm{proj}(G(1^\lambda), \mathsf{inj})\} \approx_c \{\mathrm{proj}(G(1^\lambda, \mathsf{lossy}))\}$$
3. *Lossiness of lossy keys.* If $(pk_{\mathsf{lossy}}, sk_{\mathsf{lossy}}) \xleftarrow{\$} G(1^\lambda, \mathsf{lossy})$, then for all $x_0, x_1 \in X$, the statistical distance between the distributions $E(pk_{\mathsf{lossy}}, x_0, R)$ and $E(pk_{\mathsf{lossy}}, x_1, R)$ is negligible in λ.

4. *Openability.* If $(pk_{\mathsf{lossy}}, sk_{\mathsf{lossy}}) \xleftarrow{\$} G(1^\lambda, \mathsf{lossy})$, and $r \xleftarrow{\$} \mathsf{coins}(E)$, then for all $x_0, x_1 \in X$ with overwhelming probability, there exists $r' \in \mathsf{coins}(E)$ such that $E(pk_{\mathsf{lossy}}, x_0, r) = E(pk_{\mathsf{lossy}}, x_1, r')$. In other words, there is an (unbounded) algorithm opener that can open a lossy ciphertext to *any* arbitrary plaintext with all but negligible probability.

Although openability is implied by property (3), it is convenient to state it explicitly in terms of an algorithm. In [BHY09], it was shown that, if the algorithm opener is efficient, then the encryption scheme is actually SEM-SO-ENC secure (instead of only IND-SO-ENC).

We do not explicitly require schemes to be IND-CPA secure since semantic security follows from the indistinguishability of keys and lossiness of the lossy keys. Indeed, for any $x_0, x_1 \in X$,

$$E(\mathrm{proj}(G(1^\lambda, \mathsf{inj})), x_0, R) \approx_c E(\mathrm{proj}(G(1^\lambda, \mathsf{lossy})), x_0, R)$$
$$\approx_s E(\mathrm{proj}(G(1^\lambda, \mathsf{lossy})), x_1, R) \approx_c E(\mathrm{proj}(G(1^\lambda, \mathsf{inj})), x_1, R).$$

In [BHY09], it was shown that Lossy Encryption can notably be constructed in a straightforward manner from lossy trapdoor functions. More precisely, they observed that the IND-CPA-secure system given in [PW08] is a Lossy Encryption scheme. Next, they proved the following fact.

Theorem C.2.3 [BHY09] Any Lossy Encryption scheme where the plaintext space admits a n-message sampler $\mathcal{M}$ that supports efficient resampling is IND-SO-ENC secure.

C.3 Constructing Lossy Encryption Schemes

C.3.1 Re-Randomizable Encryption Implies Lossy Encryption

In many cryptosystems, given a ciphertext c and a public-key, it is possible to re-randomize c to a new ciphertext c' such that c and c' encrypt the same plaintext but are statistically independent. We call a public key cryptosystem given by algorithms (G, E, D) *statistically re-randomizable*[3] if

- (G, E, D) is semantically-secure in the standard sense (IND-CPA).
- There is an efficient function ReRand such that if r' is chosen uniformly from $\mathsf{coins}(\mathsf{ReRand})$, and r_0 are chosen uniformly from $\mathsf{coins}(E)$, then the distributions

  $$\{r_0 \xleftarrow{\$} \mathsf{coins}(E) : E(pk, m, r_0)\} \approx_s \{r' \xleftarrow{\$} \mathsf{coins}(\mathsf{ReRand}) : \mathsf{ReRand}(E(pk, m, r_1), r')\}$$

  for all public keys pk and messages m, and randomness r_1.

There are many examples of re-randomizable encryption. For example, if (G, E, D) is *homomorphic* (*i.e.*, for any two pairs (m_0, r_0) and (m_1, r_1), we have $E(pk, m_0, r_0) \cdot E(pk, m_1, r_1) = E(pk, m_0 + m_1, r^*)$ for some $r^* \in \mathsf{coins}(E)$), it may be possible to take $\mathsf{ReRand}(pk, c, r') = c \cdot E(pk, 0, r')$. For all known homomorphic cryptosystems (such as Elgamal, Paillier, Damgård-Jurik, Goldwasser-Micali), we obtain statistically re-randomizable encryption with this definition of ReRand.

[3] We note that this definition of re-randomizable encryption requires statistical re-randomization. It is possible to define re-randomizable encryption which satisfies perfect re-randomization (stronger) or computational re-randomization (weaker). Such definitions already exist in the literature (see for example [PR07, Gro04, GJJS04, CKN03]). Our constructions require statistical re-randomization, and do not go through under a computational re-randomization assumption.

We note that, since re-randomization does not require any kind of group structure on the plaintext space or any method for combining ciphertexts, re-randomizable encryption appears to be a weaker primitive than homomorphic encryption. Although it is not implied by homomorphic encryption *per se*, all known homomorphic cryptosystems are re-randomizable. A more thorough discussion of the relationship between these primitives is given in Appendix C.B.

Our first result gives a simple and efficient method for creating lossy encryption from re-randomizable encryption. Let (G, E, D) be a statistically re-randomizable public-key cryptosystem, and we create Lossy Encryption $(\bar{G}_{\mathsf{inj}}, \bar{G}_{\mathsf{lossy}}, \bar{E}, \bar{D})$ as follows:

- **Key Generation:**
  $\bar{G}(1^\lambda, \mathsf{inj})$ generates a pair $(pk, sk) \leftarrow G(1^\lambda)$. Then $G(1^\lambda, \mathsf{inj})$ picks $r_0, r_1 \xleftarrow{\$} \mathsf{coins}(E)$, and generates $e_0 = E(pk, 0, r_0)$, $e_1 = E(pk, 1, r_1)$. $\bar{G}(1^\lambda, \mathsf{inj})$ returns $(\bar{pk}, \bar{sk}) = ((pk, e_0, e_1), sk)$.
  $\bar{G}(1^\lambda, \mathsf{lossy})$ runs $G(1^\lambda)$, generating a pair (pk, sk). Then, it picks $r_0, r_1 \xleftarrow{\$} \mathsf{coins}(E)$ and generates $e_0 = E(pk, 0, r_0)$, $e_1 = E(pk, 0, r_1)$. $\bar{G}(1^\lambda, \mathsf{lossy})$ returns $(\bar{pk}, \bar{sk}) = ((pk, e_0, e_1), sk)$.
- **Encryption:** $\bar{E}(\bar{pk}, b, r') = \mathsf{ReRand}(pk, e_b, r')$ for $b \in \{0, 1\}$.
- **Decryption** $\bar{D}(\bar{sk}, c)$, simply outputs $D(sk, c)$.

We first notice that, under an injective key, the encryption mapping is clearly injective and the decryption algorithm D performs the inverse operation. In lossy mode, it will be statistically lossy by the properties of the ReRand function. The proof that this is a Lossy Encryption system is straightforward and we check the details here.

1. *Correctness on injective keys.* This follows immediately from the correctness of E.
2. *Indistinguishability of keys.* This follows immediately from the IND-CPA security of (G, E, D).
3. *Lossiness of lossy keys.* Notice that under a lossy public-key $\bar{pk}$, e_0 and e_1 are both encryptions of zero, so that $\bar{E}(\bar{pk}, b, r)$ will also be an encryption of zero for $b \in \{0, 1\}$. By the properties of ReRand, the distributions $\{\bar{E}(\bar{pk}, 0, r)\}$ and $\{\bar{E}(\bar{pk}, 1, r)\}$ will be statistically close, which is exactly what is required for a key to be "lossy".
4. *Openability.* Under a lossy public-key, we have $\bar{E}(\bar{pk}, b, r') = \mathsf{ReRand}(E(pk, 0, r_b), r')$. Since r' is chosen uniformly from $\mathsf{coins}(\mathsf{ReRand})$, the properties of ReRand guarantee that the distributions $\mathsf{ReRand}(E(pk, 0, r_b), r')$ and $\mathsf{ReRand}(E(pk, 0, r_{1-b}), r'')$ are statistically close. The existence of r'' such that $\mathsf{ReRand}(E(pk, 0, r_b), r') = \mathsf{ReRand}(E(pk, 0, r_{1-b}), r'')$ then follows from lemma C.3.1.

Lemma C.3.1 If R is a random variable, and $f : R \to X$, $g : R \to Y$ and

$$\sum_{z \in X \cup Y} \Pr[r \leftarrow R : f(r) = z] - \Pr[r \leftarrow R : g(r) = z] = \nu,$$

then $\Pr[r \leftarrow R : \forall r' \in R, f(r) \neq g(r')] < \nu$.

Proof: It suffices to notice that

$$\begin{aligned}
\nu &= \sum_{z \in X \cup Y} \Pr[r \leftarrow R : f(r) = z] - \Pr[r \leftarrow R : g(r) = z] \\
&\geq \sum_{z \in X \setminus Y} \Pr[r \leftarrow R : f(r) = z] - \Pr[r \leftarrow R : g(r) = z] \\
&= \Pr[r \leftarrow R : \forall r' \in R, f(r) \neq g(r')] .
\end{aligned}$$

Although this scheme only allows encrypting single bits, it can be easily modified to encrypt longer messages if the underlying cryptosystem is homomorphic and if the set of encryptions of zero can be almost uniformly sampled (the details are available in Appendix C.B).

The above construction is easily seen to give a perfectly-binding SOA secure commitment scheme (with trusted setup). If our goal is only to construct SOA secure commitments, we do not need re-randomizable encryption, and a weaker primitive suffices. In Appendix C.A, we define *re-randomizable one-way functions* and show that these imply SOA secure commitments. While these constructions both require a trusted setup, in a sense, this is inevitable since it was shown in [Hof11, BHY09] that perfectly-binding SOA secure commitments without trusted setup cannot be created in a black-box manner from any primitive with a game-based definition of security.

We also note that specific homomorphic cryptosystems such as Paillier [Pai99] or Damgård-Jurik [DJ01] provide more efficient constructions where multi-bit messages can be encrypted. In addition, as shown in Appendix C.C.1, the factorization of the modulus N provides a means for efficiently opening a lossy ciphertext to any plaintext. Thus this scheme is actually SEM-SO-ENC secure when instantiated with these cryptosystems. This provides the most efficient known examples of SEM-SO-ENC secure cryptosystems. See Appendix C.C.1 for further discussion.

C.3.2 Statistically-Hiding $\binom{2}{1}$-OT Implies Lossy Encryption

We briefly recall the definition of honest-receiver two-round statistically-hiding $\binom{2}{1}$-OT. Oblivious transfer is a protocol between a sender Sen and a receiver $\mathsf{Rec} = (\mathsf{Rec}_q, \mathsf{Rec}_r)$. The sender Sen has two strings s_0, s_1, and the receiver has a bit b. The receiver Rec_q generates a query q along with some state information sk and sends q to the sender. The sender evaluates $\mathsf{q}(s_0, s_1)$ and sends the result $\mathsf{rsp} = \mathsf{Sen}(\mathsf{q}, s_0, s_1)$ to the receiver Rec_r who uses sk to obtain s_b.

- **Correctness:** For all $s_0, s_1 \in \{0,1\}^k$, for all $b \in \{0,1\}$, there is a negligible function ν such that

$$\Pr[(\mathsf{q}, sk) \xleftarrow{\$} \mathsf{Rec}_q(1^\lambda, b);\ \mathsf{rsp} \xleftarrow{\$} \mathsf{Sen}(\mathsf{q}, s_0, s_1) : \mathsf{Rec}_r(sk, \mathsf{rsp}) = s_b] \geq 1 - \nu(\lambda).$$

- **Receiver Privacy:** b remains computationally hidden from Sen's view. Specifically, we must have

$$\{(\mathsf{q}, sk) \xleftarrow{\$} \mathsf{Rec}_q(1^\lambda, 0) : \mathsf{q}\} \approx_c \{(\mathsf{q}, sk) \xleftarrow{\$} \mathsf{Rec}_q(1^\lambda, 1) : \mathsf{q}\},$$

  where the distributions are taken over the internal randomness of Rec_q.

- **Sender Privacy:** for any $b \in \{0,1\}$, for any strings s_0, s_1, s_0', s_1' such that $s_b = s_b'$ and any honest receiver's query $\mathsf{q} = \mathsf{Rec}_q(1^\lambda, b)$, it must hold that

$$\begin{aligned} & \{(\mathsf{q}, sk) \xleftarrow{\$} \mathsf{Rec}_q(1^\lambda, b); \mathsf{rsp} \xleftarrow{\$} \mathsf{Sen}(\mathsf{q}, s_0, s_1) : \mathsf{rsp}\} \\ \approx_s \quad & \{(\mathsf{q}, sk) \xleftarrow{\$} \mathsf{Rec}_q(1^\lambda, b); \mathsf{rsp} \xleftarrow{\$} \mathsf{Sen}(\mathsf{q}, s_0', s_1') : \mathsf{rsp}\} \end{aligned}$$

  where the distributions are taken over the internal randomness of Rec_q and Sen.

Let $(\mathsf{Sen}, \mathsf{Rec})$ be a two-round honest-receiver statistically-hiding $\binom{2}{1}$-OT. We construct a lossy encryption as follows:

- **Key Generation:** Define $G(1^\lambda, \mathsf{inj}) = \mathsf{Rec}_q(1^\lambda, 0)$. Set $pk = \mathsf{q}$, and $sk = sk$.
  Define $G(1^\lambda, \mathsf{lossy}) = \mathsf{Rec}_q(1^\lambda, 1)$. Set $pk = \mathsf{q}$, and $sk = \perp$.

- **Encryption:** Define $E(pk, m, (r, r^*)) = \mathsf{Sen}(\mathsf{q}, m, r; r^*)$, where r^* is the randomness used in $\mathsf{Sen}(\mathsf{q}, m, r)$ and $r \xleftarrow{\$} \{0,1\}^{|m|}$ is a random string.
- **Decryption:** to decrypt $c = \mathsf{rsp}$ in injective mode, we define $D(sk, \mathsf{rsp}) = \mathsf{Rec}_r(sk, \mathsf{rsp})$.

Lemma C.3.2 The scheme (G, E, D) forms a lossy encryption scheme.

Proof: We need to show three things:

- **Correctness on injective keys:** This follows immediately from the correctness of OT.
- **Indistinguishability of keys:** This follows immediately from the receiver privacy of OT.
- **Lossiness of lossy keys:** This will follow from the statistical sender privacy OT. More precisely, if the cryptosystem is in lossy mode, the sender privacy of OT says that for all m_0, m_1
$$\{\mathsf{Sen}(\mathsf{q}, m_0, r)\} \approx_s \{\mathsf{Sen}(\mathsf{q}, m_1, r)\},$$
where the distribution is taken over the internal randomness of Sen. Now, if we view the randomness of Sen as an explicit input to Sen (as we do in encryption), then we have that for all m_0, m_1 and r,
$$\Delta(\mathsf{Sen}(\mathsf{q}, m_0, r; \cdot), \mathsf{Sen}(\mathsf{q}, m_1, r); \cdot) < \nu,$$
where the distributions are taken over the internal randomness of Sen. Applying lemma C.3.3, we find
$$\Delta(\mathsf{Sen}(\mathsf{q}, m_0, \cdot; \cdot), \mathsf{Sen}(\mathsf{q}, m_1, \cdot; \cdot)) \leq \nu,$$
where the distributions range over the uniform choice of r and the internal randomness of Sen. This is exactly what is required to guarantee the lossiness of lossy keys.

∎

Lemma C.3.3 Let X, Y, Z be random variables such that $\Delta(X, Y|Z = z) < \epsilon$ for all z. Then, $\Delta(X, Y) < \epsilon$.

Proof:

$$\begin{aligned}
\Delta(X, Y) &= \sum_a |\Pr(X = a) - \Pr(Y = a)| \\
&= \sum_a \sum_z |\Pr(X = a, Z = z) - \Pr(Y = a, Z = z)| \\
&= \sum_a \sum_z |\Pr(X = a|Z = z) - \Pr(Y = a|Z = z)| \Pr(z = z) \\
&= \sum_z \Pr(Z = z) \sum_a |\Pr(X = a|Z = z) - \Pr(Y = a|Z = z)| \\
&= \sum_z \Pr(Z = z)\Delta(X, Y|Z = z) < \epsilon \sum_z \Pr(Z = z) = \epsilon.
\end{aligned}$$

∎

Applying the results of [DMO00] which show that single-server Private Information Retrieval (PIR) implies statistically-hiding OT, we find the following corollary.

Corollary C.3.4 One round (two message) Single-Server PIR implies Lossy-Encryption.

Since homomorphic encryption implies PIR [KO97, Man98, IKO05], the following result follows.

Corollary C.3.5 Homomorphic encryption implies Lossy-Encryption.

It was shown in [Kal05, HK12] that, in the half simulation model, statistically hiding $\binom{2}{1}$-OT can be based on smooth hash proof systems that fit a slight modification of the original definition [CS02] with suitable verifiability properties. In the honest-but-curious receiver setting (which suffices here), it was already noted in [HK12][Section 1.3] that ordinary hash proof systems, as defined in [CS02], are sufficient to realize $\binom{2}{1}$-OT. In Appendix C.D, we describe a simplification of the construction of lossy encryption from hash proof systems and obtain the next result.

Corollary C.3.6 Smooth projective hash functions imply Lossy Encryption.

Interestingly, the DDH-based lossy encryption scheme of [KN08,PVW08,BHY09] can be seen as a particular instance of that construction using the Projective Hashing of [CS98]. It can also be interpreted as being derived (after simplification) from the Naor-Pinkas OT protocol [NP01] via our construction.

The relationship with hash proof systems also suggests other implementations of lossy encryption based on Composite or Quadratic Residuosity (which differ from the scheme in Appendix C.C.1 and from Goldwasser-Micali, respectively) and the Decision Linear assumption [BBS04].

To summarize this section, by applying Theorem C.2.3, we obtain the following theorem.

Theorem C.3.7 Statistically-hiding 2-round honest-player $\binom{2}{1}$-OT implies IND-SO-ENC secure encryption. Moreover, single-server PIR and homomorphic encryption and smooth projective hash proof systems also imply IND-SO-ENC secure encryption.

C.4 Chosen-Ciphertext Security

It has long been recognized that if an adversary is given access to a decryption oracle, many cryptosystems may become insecure. The notion of chosen-ciphertext Security [NY90,RS91,DDN91] was created to address this issue, and since then there have been many schemes that achieve this level of security. The attacks of Bleichenbacher on RSA PKCS#1 [Ble98] emphasized the practical importance of security against chosen-ciphertext attacks (CCA).

The need for selective opening security was first recognized in the context of Multi-Party Computation (MPC), where an active MPC adversary can view all ciphertexts sent in a current round and then choose a subset of senders to corrupt. It is natural to imagine an adversary who, in addition to corrupting a subset of senders, can also mount a chosen-ciphertext attack against the receiver. Schemes proposed so far (based on re-randomizable encryption or described in [BHY09]) are obviously insecure in this scenario.

In this section, we extend the notion of chosen-ciphertext security to the selective opening setting. As in the standard selective-opening setting, we can define security either by indistinguishability, or by simulatability. We will give definitions of security as well as constructions for both settings.

Currently known techniques to acquire chosen-ciphertext security are delicate to use here. For instance, handling decryption queries using the Naor-Yung paradigm [NY90] and non-interactive zero-knowledge techniques [Sah99] is not straightforward as, when the adversary

makes his corruption query, he should also obtain the random coins that were used to produce NIZK proofs. Hash proof systems (HPS) [CS98, CS02] seem problematic to use as well. They typically involve security reductions where simulators know the private key corresponding to the public key given to the adversary. This seems inherently at odds with the features of lossy encryption, where security relies on the property that lossy public keys (for which private keys may not exist) look like well-formed public keys. As we will see, leveraging other tools such as the Canetti-Halevi-Katz paradigm [CHK04] raises its deal of technical issues.

C.4.1 Chosen-Ciphertext Security: Indistinguishability

We begin with the indistinguishability-based definition (the simulation-based one is provided in Appendix C.E).

We define two games, a real game (ind-cca2-real) and an ideal game (ind-cca2-ideal). In both games, the challenger runs the key-generation algorithm to generate a key pair $(sk, pk) \leftarrow G(1^\lambda)$ and sends pk to $\mathcal{A}$. The adversary is then allowed to adaptively make the following types of queries.

- **Challenge Query:** let $\mathcal{M}$ be a message sampler. The latter samples $\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$ and returns n "target" ciphertexts

$$\mathbf{C} = (\mathbf{C}[1], \ldots, \mathbf{C}[n]) \leftarrow (E(pk, m_1, r_1), \ldots, E(pk, m_n, r_n)).$$

- **Corrupt Query:** $\mathcal{A}$ chooses a subset $I \subset \{1, \ldots, n\}$ of cardinality $\#I = n/2$. The challenger then reveals $\{(m_i, r_i)\}_{i \in I}$ to $\mathcal{A}$.
  - In the real game, the challenger then sends $\{m_j\}_{j \notin I}$ to the adversary.
  - In the ideal game, the challenger re-samples $\mathbf{m}' = (m'_1, \ldots, m'_n) \xleftarrow{\$} \mathcal{M}_{|I,\mathbf{m}[I]}$ (*i.e.*, in such a way that $m'_j = m_j$ for each $j \in I$) and sends $\{m'_j\}_{j \notin I}$ to $\mathcal{A}$.
- **Decryption Queries:** $\mathcal{A}$ chooses a ciphertext C that has never appeared as a target ciphertext and sends C to the challenger which responds with $D(sk, C)$.

After a polynomial number of queries, exactly one of which is a challenge query and precedes the corrupt query (which is unique as well), the adversary outputs $b \in \{0, 1\}$.

Definition C.4.1 A public key cryptosystem is IND-SO-CCA2 secure if, for any polynomial n and any n-message sampler $\mathcal{M}$ supporting efficient conditional re-sampling, any PPT adversary $\mathcal{A}$ has negligibly different outputs in the real game and in the ideal game: for some negligible function ν, we must have

$$\left|\Pr[\mathcal{A}^{\mathsf{ind\text{-}cca2\text{-}real}} = 1] - \Pr[\mathcal{A}^{\mathsf{ind\text{-}cca2\text{-}ideal}} = 1]\right| < \nu.$$

If the adversary is not allowed to make decryption queries, this reduces to IND-SO-ENC security.

Our construction of IND-SO-CCA2 secure encryption requires some basic tools outlined below.

C.4.2 Chameleon Hash Functions

A chameleon hash function [KR00] $\mathcal{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ consists of a key generation algorithm CMKg that, given a security parameter λ, outputs a pair $(hk, tk) \xleftarrow{\$} \mathcal{G}(\lambda)$. The randomized hashing algorithm outputs $y = \mathsf{CMhash}(hk, m, r)$ given the public key hk, a message

m and random coins $r \in \mathcal{R}_{hash}$. On input of m, r, m' and the trapdoor key tk, the switching algorithm $r' \leftarrow \mathsf{CMswitch}(tk, m, r, m')$ outputs $r' \in \mathcal{R}_{hash}$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$. Collision-resistance mandates that it be infeasible to find collisions (*i.e.*, pairs $(m', r') \neq (m, r)$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$) without knowing tk. Finally, uniformity guarantees that the distribution of hashes is independent of the message m, in particular, for all hk, and m, m', the distributions $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m, r)\}$ and $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m', r)\}$ are identical. It is well-known that chameleon hashing can be based on standard number theoretic assumptions such as factoring or the discrete logarithm.

C.4.3 A Special Use of the Canetti-Halevi-Katz Paradigm

The Canetti-Halevi-Katz technique [CHK04] is a method to build chosen-ciphertext secure encryption schemes from weakly secure identity-based or tag-based encryption scheme. A tag-based encryption scheme (TBE) [MRY04, Kil06] is a public key cryptosystem where the encryption and decryption algorithms take an additional input, named the *tag*, which is a binary string of appropriate length with no particular structure. A TBE scheme consists of a triple $\mathcal{TBE} = (\mathsf{TBEKg}, \mathsf{TBEEnc}, \mathsf{TBEDec})$ of efficient algorithms where, on input of a security parameter λ, TBEKg outputs a private/public key pair (pk, sk); TBEEnc is a randomized algorithm that outputs a ciphertext C on input of a public key pk, a string θ – called *tag* – and a message $m \in \mathsf{MsgSp}(\lambda)$; $\mathsf{TBEDec}(sk, \theta, C)$ is the decryption algorithm that takes as input a secret key sk, a tag θ and a ciphertext C and returns a plaintext m or $\perp$. Associated with $\mathcal{TBE}$ is a plaintext space MsgSp. Correctness requires that for all $\lambda \in \mathbb{N}$, all key pairs $(pk, sk) \leftarrow \mathsf{TBEKg}(1^\lambda)$, all tags θ and any plaintext $m \in \mathsf{MsgSp}(\lambda)$, it holds that $\mathsf{TBEDec}(sk, \theta, \mathsf{TBEEnc}(pk, \theta, M)) = m$.

Selective Opening Security for TBE Schemes. In the selective opening setting, the weak CCA2 security definition of [Kil06] can be extended as follows.

Definition C.4.2 A TBE scheme $\mathcal{TBE} = (\mathsf{TBEKg}, \mathsf{TBEEnc}, \mathsf{TBEDec})$ is *selective-tag weakly IND-SO-CCA2 secure* (or IND-SO-stag-wCCA2 secure) if, for any polynomial n and any n-message sampler $\mathcal{M}$ supporting efficient conditional re-sampling, any PPT adversary $\mathcal{A}$ produces negligibly different outputs in the real and ideal games, which are defined as follows.

1. The adversary $\mathcal{A}$ chooses n tags $\theta_1^\star, \ldots, \theta_n^\star$ and sends them to the challenger.
2. The challenger generates a key pair $(sk, pk) \leftarrow \mathsf{TKEKg}(1^\lambda)$ and hands pk to $\mathcal{A}$. The latter then adaptively makes the following kinds of queries:
   - **Challenge Query:** let $\mathcal{M}$ be a message sampler for $\mathsf{MsgSp}(\lambda)$. The challenger samples $\mathbf{m} = (m_1, \ldots, m_n) \xleftarrow{\$} \mathcal{M}$ and returns n target ciphertexts
     $$\mathbf{C} = (\mathbf{C}[1], \ldots, \mathbf{C}[n]) \leftarrow (\mathsf{TBEEnc}(pk, \theta_1^\star, m_1, r_1), \ldots, \mathsf{TBEEnc}(pk, \theta_n^\star, m_n, r_n)).$$
   - **Corrupt Query:** $\mathcal{A}$ chooses a subset $I \subset \{1, \ldots, n\}$ of size $\#I = n/2$. The challenger then hands $\{(m_i, r_i)\}_{i \in I}$ to $\mathcal{A}$.
     - In the real game, the challenger then sends $\{m_j\}_{j \notin I}$ to the adversary.
     - In the ideal game, the challenger re-samples $(m_1', \ldots, m_n') \xleftarrow{\$} \mathcal{M}_{|I, \mathbf{m}[I]}$ and reveals $\{m_j'\}_{j \notin I}$.
   - **Decryption Queries:** $\mathcal{A}$ sends a pair (C, θ) such that $\theta \notin \{\theta_1^\star, \ldots, \theta_n^\star\}$. The challenger replies with $\mathsf{TBEDec}(sk, \theta, C) \in \mathsf{MsgSp}(\lambda) \cup \{\perp\}$.

After polynomially-many queries, one of which being a challenge query, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$. His advantage $\mathbf{Adv}_{\mathcal{A}}^{\text{IND-SO-stag-wCCA2}}(\lambda)$ is defined analogously to definition C.4.1.

At first glance, one may hope to simply obtain IND-SO-CCA2 security by applying the CHK method [CHK04] to any IBE/TBE scheme satisfying some weaker level of selective opening security.

Let us assume a TBE scheme $\mathcal{TBE} = (\mathsf{TBEKg}, \mathsf{TBEEnc}, \mathsf{TBEDec})$ that is secure in the sense of definition C.4.2 and let $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature. The black-box CHK technique turns $\mathcal{TBE}$ into a public key cryptosystem $\mathcal{PKE} = (G, E, D)$ which is obtained by letting $G(1^\lambda)$ output $(sk', (\Sigma, pk'))$ where $(sk', pk') \leftarrow \mathsf{TBEKg}(1^\lambda)$. To encrypt a message m, E generates a one-time signature key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(1^\lambda)$, computes $C_{tbe} = \mathsf{TBEEnc}(pk, \mathsf{VK}, m)$ under the tag VK and sets the $\mathcal{PKE}$ ciphertext as $(\mathsf{VK}, C_{tbe}, \sigma)$, where $\sigma = \mathcal{S}(\mathsf{SK}, C_{tbe})$.

When we try to use this transformation in the selective opening setting, the problem is that, when the adversary makes his corruption query in the reduction, he must also obtain the random coins that were used to generate one-time signature key pairs appearing target ciphertexts. Then, he is able to re-compute the corresponding one-time private keys and make decryption queries for ciphertexts involving the same verification keys as target ciphertexts, which causes the reduction to fail. Although schemes using one-time signatures do not appear to become trivially insecure, the reduction of [CHK04, Kil06] ceases to go through and the same hurdle arises with the Boneh-Katz transformation [BK05].

It was showed in [Zha07] that chameleon hash functions [KR00] can be used to turn certain TBE schemes, termed *separable*, into full-fledged IND-CCA2 cryptosytems and supersede one-time signatures in the CHK transform. A TBE scheme is said *separable* if, on input of pk, m, θ, the encryption algorithm $\mathsf{TBEEnc}(pk, t, m)$ uses randomness $r \in \mathcal{R}_{tbe}$ and returns $C_{tbe} = (f_1(pk, m, r), f_2(pk, r), f_3(pk, \theta, r))$, where functions f_1, f_2 and f_3 are computed independently of each other and are all deterministic (and give the same outputs when queried twice on the same (m, r), r and (θ, r)).

The construction of [Zha07] uses chameleon hashing instead of one-time signatures. Key generation requires to create a TBE key pair (pk', sk') and a chameleon hashing public key hk. The private key of $\mathcal{PKE}$ is the TBE private key sk'. Encryption and decryption procedures are depicted on figure C.3.

$E(m, pk)$	$D(sk, C)$
Parse pk as (pk', hk)	Parse C as (u, v, w, r_2) and sk as sk'
$r_1 \leftarrow \mathcal{R}_{tbe}$; $r_2 \leftarrow \mathcal{R}_{hash}$	$\theta = \mathsf{CMhash}(hk, u \Vert v, r_2)$
$u = f_1(pk', m, r_1)$; $v = f_2(pk', r_1)$	Return $m \leftarrow \mathsf{TBEDec}(sk', \theta, (u, v, w))$
$\theta = \mathsf{CMhash}(hk, u \Vert v, r_2)$	
$w = f_3(pk', \theta, r_1)$	
Return $C = (u, v, w, r_2)$	

Figure C.3: The Separable-TBE-to-PKE transform

Unlike the fully black-box transform where tags are generated independently of the TBE ciphertext, this construction computes the ciphertext without using any other secret random coins than those of the underlying TBE ciphertext. The tag is derived from a ciphertext component u and some independent randomness r_2 that *publicly* appears in the ciphertext. For this reason, we can hope to avoid the difficulty that appears with the original CHK transform. We prove that it is indeed the case and that any separable TBE that satisfies definition C.4.2 yields an IND-SO-CCA2 encryption scheme.

Theorem C.4.3 If $\mathcal{TBE} = (\mathsf{TBEKg}, \mathsf{TBEEnc}, \mathsf{TBEDec})$ is a separable TBE scheme with IND-SO-stag-wCCA2 security, the transformation of figure C.3 gives an IND-SO-CCA2 PKE scheme. For any IND-SO-CCA2 adversary $\mathcal{A}$, there is a TBE adversary $\mathcal{A}^{tbe}$ and a chameleon hash

adversary $\mathcal{A}^{hash}$ s.t.

$$\mathbf{Adv}_{\mathcal{A}}^{\text{IND-SO-CCA2}}(\lambda) \leq 2 \cdot (\mathbf{Adv}_{\mathcal{A}^{tbe}}^{\text{IND-SO-stag-wCCA2}}(\lambda) + qn\delta + \mathbf{Adv}_{\mathcal{A}^{hash}}^{\text{CR-CMhash}}(\lambda)),$$

where q is the number of decryption queries and δ is the maximal probability, taken over the random choice of $r_1 \in \mathcal{R}_{tbe}$, that f_2 outputs a specific element of its range.

Proof: We first note that the definition of IND-SO-CCA2 security is equivalent to a definition where the adversary $\mathcal{A}$ is faced with a simulator and has to decide whether the latter is playing the real game, where the actual plaintexts are revealed after the corruption query, or the ideal game. The game to be played is determined by a random bit $b \in \{0,1\}$ secretly chosen by the challenger and which $\mathcal{A}$ has to guess.
Using this definition, the proof is similar to [Zha07] and considers two kinds of adversaries.

- Type I attackers never invoke the decryption oracle on (u, v, w, r_2) for which the hash value $\mathsf{CMhash}(hk, u||v, r_2)$ collides with a tags $\theta_i^\star$ associated with target ciphertexts.
- Type II adversaries make at least one decryption query for a valid ciphertext (u, v, w, r_2) such that $\mathsf{CMhash}(hk, u||v, r_2)$ hits the tag $\theta_i^\star$ of some target ciphertext.

Type I adversaries are handled similarly to [Zha07]. We outline an adversary $\mathcal{A}^{tbe}$ against the TBE scheme using a type I IND-SO-CCA2 adversary $\mathcal{A}$. The former begins by generating a key pair $(hk, tk) \leftarrow \mathsf{CMhash}(\lambda)$ for the chameleon hash. It chooses dummy $u_i', v_i', r_{2,i}'$ in the appropriate domains and uses them to generate tags $\theta_i^\star = \mathsf{CMhash}(hk, u_i'||v_i', r_{2,i}')$ for $i = 1, \ldots, n$. These are transmitted to $\mathcal{A}^{tbe}$'s challenger $\mathcal{C}$, which replies with a TBE public key pk'. The public key $pk = (pk', hk)$ is given to $\mathcal{A}$.
Any decryption query made by $\mathcal{A}$ is forwarded to $\mathcal{A}^{tbe}$'s challenger $\mathcal{C}$ and the latter's response is relayed to $\mathcal{A}$. When $\mathcal{A}$ outputs a plaintext distribution $\mathcal{M}$, $\mathcal{A}^{tbe}$ sends $\mathcal{M}$ to his own challenger. Upon receiving the vector of target ciphertexts $\mathbf{C}_{tbe}^\star = (C_{tbe}[1]^\star, \ldots, C_{tbe}[n]^\star)$ (where $C_{tbe}[i]^\star = (u_i^\star, v_i^\star, w_i^\star)$ is associated with the tag $\theta_i^\star$), $\mathcal{A}^{tbe}$ uses the trapdoor tk to compute $r_{2,i}^\star = \mathsf{CMswitch}(tk, u_i'||v_i', r_{2,i}', u_i^\star||v_i^\star)$ (so that $\theta_i^\star = \mathsf{CMhash}(hk, u_i^\star||v_i^\star, r_{2,i}^\star) = \mathsf{CMhash}(hk, u_i'||v_i', r_{2,i}')$) and sends the target vector $\mathbf{C}^\star = (\mathbf{C}[1]^\star, \ldots, \mathbf{C}[n]^\star)$, where $\mathbf{C}[i]^\star = (u_i^\star, v_i^\star, w_i^\star, r_{2,i}^\star)$ for all i, to $\mathcal{A}$.
Then, $\mathcal{A}$ makes new decryption queries, which $\mathcal{A}^{tbe}$ handles by simply transmitting them to $\mathcal{C}$ and relaying the latter's responses back to $\mathcal{A}$. When $\mathcal{A}$ decides to make his corruption query $I \subset \{1, \ldots, n\}$, $\mathcal{A}^{tbe}$ sends I to $\mathcal{C}$ that replies with plaintexts and random coins $\{(m_i^\star, r_{1,i}^\star)\}_{i \in I}$ for ciphertexts $\{\mathbf{C}_{tbe}[i]^\star\}_{i \in I}$ as well as $\{m_i\}_{i \notin I}$ for which $\mathcal{A}^{tbe}$ aims at deciding whether $m_i = m_i^\star$ for all i or $m_i \in_R \mathcal{M}$. All these elements are passed to $\mathcal{A}$ (note that $\mathcal{A}^{tbe}$ does not need to include $\{r_{2,i}^\star\}_{i \in I}$ as $\mathcal{A}$ already obtained them as part of $\mathbf{C}[i]^\star$) who makes new decryption queries.
Since $\mathcal{A}$ is assumed to be a Type I adversary, no such decryption query (u, v, w, r_2) ever results in a tag $\theta = \mathsf{CMhash}(hk, u||v, r_2)$ such that $\theta \in \{\theta_1^\star, \ldots, \theta_n^\star\}$, $\mathcal{A}^{tbe}$ can always query $\mathcal{C}$ to decrypt $((u, v, w), \theta)$ and give the answer back to $\mathcal{A}$. Eventually, $\mathcal{A}^{tbe}$ outputs the same result $b' \in \{0,1\}$ as $\mathcal{A}$ and we easily see that, if $\mathcal{A}$ is successful, so is $\mathcal{A}^{tbe}$. Therefore, it comes that $\mathbf{Adv}^{\text{Type-I}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-SO-stag-wCCA2}}(\mathcal{A}^{tbe})$.

Type II adversaries. In the expectation of a Type II adversary, we construct a collision-finder $\mathcal{A}^{hash}$ that sets up a public key (pk', hk) by obtaining the chameleon hash key hk from a challenger and generates $(sk', pk') \leftarrow \mathsf{TBEKg}(\lambda)$ on its own. It challenges the adversary $\mathcal{A}$ on the public key $pk = (pk', hk)$ and uses the private key sk' to perfectly handle all decryption queries. At the challenge step, $\mathcal{A}$ outputs a distribution $\mathcal{M}$ and obtains a vector $\mathbf{C}^\star = (\mathbf{C}[1]^\star, \ldots, \mathbf{C}[n]^\star)$ of target ciphertexts, where, for each $i \in \{1, \ldots, n\}$, $\mathbf{C}[i]^\star =$

$(u_i^\star, v_i^\star, w_i^\star, r_{2,i}^\star)$ with $u_i^\star = f_1(pk, m_i^\star, r_{1,i}^\star)$, $v_i^\star = f_2(pk, r_{1,i}^\star)$, $\theta_i^\star = \mathsf{CMhash}(hk, u_i^\star||v_i^\star, r_{2,i}^\star)$ and $w_i^\star = f_3(pk, \theta_i^\star, r_{1,i}^\star)$ for plaintexts $m_i^\star \xleftarrow{\$} \mathcal{M}$ and random values $r_{1,i}^\star \xleftarrow{\$} \mathcal{R}_{tbe}, r_{2,i}^\star \xleftarrow{\$} \mathcal{R}_{hash}$.
In the simulation, algorithm $\mathcal{A}^{hash}$ aborts and fails in the event that, for some index $i \in \{1, \ldots, n\}$, the ciphertext $\mathbf{C}[i]^\star = (u_i^\star, v_i^\star, w_i^\star, r_{2,i}^\star)$ is such that $v_i^\star$ previously appeared in a decryption query. This only occurs with probability smaller than $qn\delta$ if δ denotes the maximal probability, taken over the random choice of $r_{1,i}^\star \xleftarrow{\$} \mathcal{R}_{tbe}$, that a specific element of the image of f_2 is reached.
If $\mathcal{A}^{hash}$ does not abort, $\mathcal{A}$ makes new decryption queries that $\mathcal{A}^{hash}$ still perfectly answers using sk'. At some point, $\mathcal{A}$ makes a corruption query I and obtains $\{(m_i^\star, r_{1,i}^\star, r_{2,i}^\star)\}_{i \in I}$. Plaintexts $\{m_i\}_{i \notin I}$ are the actual plaintexts if the challenger $\mathcal{A}^{hash}$'s random bit is $b = 0$ and random plaintexts if $b = 1$.
$\mathcal{A}$ is assumed to query at some point the decryption of some ciphertext $C = (u, v, w, r_2)$ such that $\theta = \mathsf{CMhash}(hk, u||v, r_2) = \mathsf{CMhash}(hk, u_i^\star||v_i^\star, r_{2,i}^\star) = \theta_i^\star$ for some $i \in \{1, \ldots, n\}$. If that query is made before the challenge phase, we must have $v \neq v_i^\star$ as $\mathcal{A}^{hash}$ would have aborted in the challenge phase otherwise. If the query is a post-challenge query, we also have $(u, v, r_2) \neq (u_i^\star, v_i^\star, r_{2,i}^\star)$ since, for any valid ciphertext, $(u, v) = (u_i^\star, v_i^\star)$ and $\theta = \theta_i^\star$ would imply $w = w_i^\star$ and C would be a target ciphertext. In either case, we have a collision on the chameleon hash.
The above arguments give us the upper bound $\mathbf{Adv}^{\text{Type-II}}(\mathcal{A}) \leq qn\delta + \mathbf{Adv}^{\text{CR-CMhash}}(\mathcal{A}^{hash})$.

The theorem is established by noting that $\mathcal{A}^{hash}$ can guess upfront (by flipping a coin independently of $\mathcal{A}$'s view) which kind of attack the adversary will mount and prepare the public key accordingly.

∎

C.4.4 Lossy and All-But-n Trapdoor Functions

Lossy trapdoor functions were first defined in [PW08]. A tuple $(S_{\text{ltdf}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$ of PPT algorithms is called a family of (d, k)-lossy trapdoor functions if the following properties hold:

- **Sampling injective functions:** $S_{\text{ltdf}}(1^\lambda, 1)$ outputs (s, t), where s is a function index and t its trapdoor. It is required that $F_{\text{ltdf}}(s, \cdot)$ be injective on $\{0,1\}^d$ and $F_{\text{ltdf}}^{-1}(t, F_{\text{ltdf}}(s, x)) = x$ for all x.

- **Sampling lossy functions:** $S_{\text{ltdf}}(1^\lambda, 0)$ outputs $(s, \perp)$ where s is a function index and $F_{\text{ltdf}}(s, \cdot)$ is a function on $\{0,1\}^d$, where the image of $F_{\text{ltdf}}(s, \cdot)$ has size at most 2^{d-k}.

- **Indistinguishability:** we have $\{(s, t) \xleftarrow{\$} S_{\text{ltdf}}(1^\lambda, 1) : s\} \approx_c \{(s, \perp) \xleftarrow{\$} S_{\text{ltdf}}(1^\lambda, 0) : s\}$.

Along with lossy trapdoor functions, Peikert and Waters [PW08] defined all-but-one (ABO) functions. Essentially, these are lossy trapdoor functions, except instead of having two branches (a lossy branch and an injective branch) they have many branches, all but one of which are injective.

The Peikert-Waters cryptosystem only requires such function families to have one lossy branch because a single challenge ciphertext must be evaluated (on a lossy branch) in the CCA2 game. Since the IND-SO-CCA security game involves $n > 1$ challenge ciphertexts, we need to generalize ABO functions into all-but-n (ABN) functions that have multiple lossy branches and where all branches except the specified ones are injective. In the case $n = 1$, ABN functions obviously boil down to ABO functions.

- **Sampling with a given lossy set:** For any n-subset $I \subset \mathcal{B}$, $S_{\mathrm{abn}}(1^\lambda, I)$ outputs s, t where s is a function index, and t its trapdoor. We require that for any $b \in \mathcal{B} \setminus I$, $G_{\mathrm{abo}}(s, b, \cdot)$ is an injective deterministic function on $\{0,1\}^d$, and $G^{-1}_{\mathrm{abn}}(t, b, G_{\mathrm{abn}}(s, b, x)) = x$ for all x. Additionally, for each $b \in I$, the image $G_{\mathrm{abn}}(s, b, \cdot)$ has size at most 2^{d-k}.

- **Hidden lossy sets:** For any distinct n-subsets $I_0^\star, I_1^\star \subset \mathcal{B}$, the first outputs of $S_{\mathrm{abn}}(1^\lambda, I_0^\star)$ and $S_{\mathrm{abn}}(1^\lambda, I_1^\star)$ are computationally indistinguishable.

Just as ABO functions can be obtained from lossy trapdoor functions [PW08], ABN functions can also be constructed generically from LTDFs.The recent results of Hofheinz [Hof12], show how to create All-But-Many Lossy Functions, which are Lossy Trapdoor Functions with a super-polynomial number of lossy branches. The advantage of his construction is that the description of the function is independent of N. Hofheinz's All-But-Many functions can be plugged into our constructions to shrink the size of the public-key in our constructions (see [Hof12] for details).

C.4.5 All-But-n Functions from Lossy Trapdoor Functions

Given a set $I \subset \mathcal{B}$, we create an unduplicatable set selector $\mathfrak{g} : \mathcal{B} \to \hat{\mathcal{B}}$. For each $\hat{b} \in \hat{\mathcal{B}}$, we will associate a lossy trapdoor function. Let $\hat{I} = \bigcup_{i \in I} \mathfrak{g}(i)$. For each $\hat{i} \in \hat{I}$, we will create a LTDF in lossy mode, and for each $\hat{b} \in \hat{\mathcal{B}} \setminus \hat{I}$, we will associate a LTDF in injective mode.

- **Sampling with a given lossy set:** Create an $(n, \lceil \log |\mathcal{B}| \rceil)$ unduplicatable set selector $\mathfrak{g}$. Suppose $\mathcal{B} \subset \{0,1\}^v$, then the construction outlined above produces $\mathfrak{g}$ which maps $\{0,1\}^v$ to subsets of $\mathbb{F}_\ell \times \mathbb{F}_\ell$, where $\ell = 2^{\lceil \log_2 2nv \rceil}$. For each element in $\mathbb{F}_\ell \times \mathbb{F}_\ell$, we will associate a lossy trapdoor function. Let $\hat{I} = \bigcup_{i \in I} \mathfrak{g}(i) \subset \mathbb{F}_\ell \times \mathbb{F}_\ell$. For each $y \in \hat{I}$ let F_y be an LTDF in lossy mode, and for each $y \in \mathbb{F}_\ell \times \mathbb{F}_\ell \setminus \hat{I}$, let F_y be an LTDF in injective mode.

  Now, define $G_{\mathrm{abn}}(b, x) = (F_{y_1}(x), \ldots, F_{y_\ell}(x))_{y_i \in \mathfrak{g}(b)}$.

Notice that if any of the functions F_y are injective, then G_{abn} is also injective, and if the image size of F in lossy mode is 2^r, then the images size of G_{abn} on a lossy branch is $2^{r\ell}$. Finally, we notice that the lossy set is hidden by the indistinguishability of modes of the LTDF.

This construction is generic but suffers from a lack of efficiency since the description of the function and its output both have a size growing as a function of n, which is obviously not a desirable property. Luckily for specific lossy trapdoor functions, the growth of the output size can be avoided.

C.4.6 An IND-SO-stag-wCCA2 TBE Construction

We now give a method for constructing IND-SO-stag-wCCA2 tag-based cryptosystems from lossy trapdoor functions. Using a chameleon hash function $(\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ where CMhash ranges over the set of branches $\mathcal{B}$ of the ABN family, we eventually obtain an IND-SO-CCA2 public key encryption scheme. The LTDF-based construction (and its proof) mimics the one [PW08] (in its IND-CCA1 variant).

Let $(S_{\mathrm{ltdf}}, F_{\mathrm{ltdf}}, F^{-1}_{\mathrm{ltdf}})$ be a family of (d, k)-lossy-trapdoor functions, and let $(S_{\mathrm{abn}}, G_{\mathrm{abn}}, G^{-1}_{\mathrm{abn}})$ be a family of (d, k') all-but-n functions with branch set $\{0,1\}^v$ where v is the length of a verification key for our one-time signature scheme. We require that $2d - k - k' \leq t - \kappa$, for $\kappa = \kappa(t) = \omega(\log t)$. Let $\mathcal{H}$ be a pairwise independent hash family from $\{0,1\}^d \to \{0,1\}^\ell$, with $0 < \ell < \kappa - 2\log(1/\nu)$, for some negligible $\nu = \nu(\lambda)$. The message space will be $\mathsf{MsgSp} = \{0,1\}^\ell$.

- $\mathsf{TBEKg}(1^\lambda)$: choose a random member $h \leftarrow \mathcal{H}$ of the pairwise independent hash family and generate

$$(s,t) \leftarrow S_{\mathrm{ltdf}}(1^\lambda, inj), \quad (s',t') \leftarrow S_{\mathrm{abn}}(1^\lambda, \{0,1,\ldots,n-1\}).$$

  The public key will be $pk = (s, s', h)$ and the secret key will be $sk = (t, t')$.

- $\mathsf{TBEEnc}(m, pk, \theta)$: to encrypt $m \in \{0,1\}^\ell$ under the tag $\theta \in \mathcal{B}$, choose $x \xleftarrow{\$} \{0,1\}^d$. Compute $c_0 = h(x) \oplus m$, $c_1 = F_{\mathrm{ltdf}}(s,x)$ and $c_2 = G_{\mathrm{abn}}(s,\theta,x)$ and set the TBE ciphertext as

$$C = (c_0, c_1, c_2) = \big(h(x) \oplus m,\ F_{\mathrm{ltdf}}(s,x),\ G_{\mathrm{abn}}(s',\theta,x)\big).$$

- $\mathsf{TBEDec}(C, sk, \theta)$: given $C = (c_0, c_1, c_2)$ and $sk = t$, compute $x = F^{-1}_{\mathrm{ltdf}}(t, c_1)$ and check whether $G_{\mathrm{abn}}(s,\theta,x) = c_2$. If not, output $\perp$. Otherwise, output $m = c_0 \oplus h(x)$.

The scheme is easily seen to be separable since C is obtained as $c_0 = f_1(pk, m, x) = m \oplus h(x)$, $c_1 = f_2(pk, x) = F_{\mathrm{ltdf}}(s,x)$ and $c_2 = f_3(pk, \theta, x) = G_{\mathrm{abn}}(s', \theta, x)$.

Theorem C.4.4 The algorithms described above form an IND-SO-stag-wCCA2 secure tag-based cryptosystem assuming the security of the lossy and all-but-n families.

Proof: The correctness of the scheme is clear, so we focus on the security. We prove security through a sequence of games which is close to the one of [PW08, Theorem 4.2].

Let Game_0 be the real IND-SO-stag-wCCA2 game. In this game, the adversary $\mathcal{A}$ first chooses a set of tags $\{\theta_1^\star, \ldots, \theta_n^\star\}$ under which target ciphertexts will be encrypted in the challenge phase. Recall that $\mathcal{A}$ is not allowed to query the decryption oracle w.r.t. a tag $\theta \in \{\theta_1^\star, \ldots, \theta_n^\star\}$ at any time.

Let Game_1 be identical to Game_0 except that we set the lossy branches of the all-but-n function G_{abn} to be those identified by $\{\theta_1^\star, \ldots, \theta_n^\star\}$.

Let Game_2 be identical to Game_1 except that, in the decryption algorithm, we use G^{-1}_{abn} to decrypt instead of F^{-1}_{ltdf}, *i.e.*, we set $x = G^{-1}_{\mathrm{abn}}(t', \theta, c_2)$ instead of $x = F^{-1}_{\mathrm{ltdf}}(t, c_1)$.

Let Game_3 be identical to Game_2 except that we replace the injective function with a lossy one, *i.e.*, during key-generation we generate $(s, \perp) \leftarrow S_{\mathrm{ltdf}}(1^\lambda, lossy)$, instead of $(s,t) \leftarrow S_{\mathrm{ltdf}}(1^\lambda, inj)$.

- Game_1 and Game_0 are indistinguishable by the indistinguishability of lossy sets in ABN functions.
- Game_2 does not affect $\mathcal{A}$'s view since he never makes a decryption query on a lossy-branch of G_{abn}.
- The indistinguishability of Game_3 and Game_2 follows from the indistinguishability of lossy and injective modes of lossy-trapdoor functions.

Now, if we can show that an adversary's probability of success in Game_3 is negligible, we will be done. To this end, we follow the proof that Lossy Encryption is selective opening secure and apply Theorem C.A.5 in [BHY09]. The key observation is that in Game_3, the challenge ciphertexts are *statistically* independent of the underlying messages. We begin by showing that this is, in fact, the case.

Now, $F_{\mathrm{ltdf}}(s,\cdot)$ and $G_{\mathrm{abn}}(s',\theta_i^\star,\cdot)$ are lossy functions with image sizes at most 2^{d-k} and $2^{d-k'}$ respectively for each $i \in [n]$. Thus the function $x \mapsto (F_{\mathrm{ltdf}}(s,x), G_{\mathrm{abn}}(s',\theta_i^\star,x))$ takes on at most $2^{2d-k-k'} \leq 2^{d-\kappa}$ values. Now by Lemma 2.1 of [PW08], the average min-entropy is bounded below

$$\tilde{H}_\infty(x|c_1,c_2,s,s') \geq H_\infty(x|s,s') - (d-\kappa) = t - (d-\kappa) = \kappa.$$

Since $\ell \leq \kappa - 2\log(1/\nu)$, by Lemma 2.2 of [PW08], for each target ciphertext $C = (c_0,c_1,c_2)$, we have

$$\Delta((c_1,c_2,h,h(x)),(c_1,c_2,h,U_\ell)) \leq \nu,$$

where U_ℓ stands for the uniform distribution on $\{0,1\}^\ell$. Now, we can incorporate the ideas of Theorem C.A.5. Since the target ciphertexts are statistically independent of the underlying plaintexts, there is a (possibly inefficient)algorithm opener, which, given (c_0,c_1,c_2,m) outputs x such that $F_{\mathrm{ltdf}}(s,x) = c_1$, $G_{\mathrm{abn}}(s,\theta_i^\star,x) = c_2$, and $h(x) \oplus m = c_0$. If no such x exists, opener outputs $\perp$ (the statistical closeness guarantees that this happens with probability at most ν).

Now, let us consider a new series of games. Let Game_{3_0} be identical to Game_3, except that target ciphertexts are opened using the output of opener instead of the actual randomness used by the challenger.

Now, for $j \in [n]$, let Game_{3_j} be identical to Game_{3_0} except that for $i \leq j$, the target ciphertexts are

$$(E(pk,\xi,r_1),\ldots,E(pk,\xi,r_j),E(pk,m_{j+1},r_{j+1}),\ldots,E(pk,m_n,r_n))$$

So, the only difference between Game_{3_j} and $\mathrm{Game}_{3_{j-1}}$ lies in whether the j^{th} target ciphertext is an encryption of a dummy message ξ or m_j. Since these two distributions are *statistically* close, even an *unbounded* adversary has a negligible chance of distinguishing them. Thus by the triangle inequality, an unbounded adversary has a negligible probability of distinguishing Game_{3_0} from Game_{3_n}.

But Game_{3_n} is identical in both the real and ideal games, so an adversary has at most a negligible probability of distinguishing the two worlds. ∎

When the scheme is instantiated with the lossy TDF of [RS09,BFO08] and the ABN function of section C.4.7, the proof of the above theorem can be adapted as follows. We simply introduce an intermediate game between Game_1 and Game_2 and consider a failure event which reveals a non-trivial factor of the modulus N if it occurs. In this game, ciphertexts are still decrypted via F_{ltdf}^{-1} and the trapdoor of the ABN function is not used. Suppose that the adversary $\mathcal{A}$ makes a decryption query involving a tag θ such that $\gcd(P(\theta),N) \neq 1$, where $P(\theta) = \prod_{i=1}^n (\theta - \theta_i^\star)$. Since $N > 2^\lambda$ and $\theta_i^\star \in \{0,1\}^\lambda$ for each tag $\theta_i^\star$, we cannot have $\theta = \theta_i^\star \bmod N$ for any $i \in \{1,\ldots,n\}$ since it would imply $\theta = \theta_i^\star$ (which is forbidden by the IND-stag-wCCA2 rules). Hence, the failure event would imply $p|(\theta - \theta_i^\star)$ and $q|(\theta - \theta_j^\star)$ for *distinct* $i,j \in \{1,\ldots,n\}$, which would reveal a non-trivial factor of N and *a fortiori* break the DCR assumption.

C.4.7 An All-but-n Function with Short Outputs

While generic, the all-but-n function of Section C.4.5 has the disadvantage of long outputs, the size of which is proportional to nk. Efficient lossy and all-but-one functions can be based on the Composite Residuosity assumption [RS09,BFO08] and the Damgård-Jurik cryptosystem [DJ01]. We show that the all-but-one function of [RS09,BFO08] extends into an all-but-n function that retains short (*i.e.*, independent of n or k) outputs. Multiple lossy branches can be obtained using a technique that traces back to the work of Chatterjee and Sarkar [CS06] who used it in the context of identity-based encryption.

- **Sampling with a given lossy set:** given a security parameter $\lambda \in \mathbb{N}$ and the desired lossy set $I = \{\theta_1^\star, \ldots, \theta_n^\star\}$, where $\theta_i^\star \in \{0,1\}^\lambda$ for each $i \in \{1, \ldots, n\}$, let $\gamma \geq 4$ be a polynomial in λ.

  1. Choose random primes p, q s.t. $N = pq > 2^\lambda$.
  2. Generate a vector $\vec{U} \in (\mathbb{Z}^*_{N^{\gamma+1}})^{n+1}$ as follows. Let $\alpha_{n-1}, \ldots, \alpha_0 \in \mathbb{Z}_{N^\gamma}$ be coefficients obtained by expanding $P[T] = (T-\theta_1^\star)\cdots(T-\theta_n^\star) = T^n + \alpha_{n-1}T^{n-1} + \cdots + \alpha_1 T + \alpha_0$ in $\mathbb{Z}_{N^\gamma}[T]$ (note that $P[T]$ is expanded in $\mathbb{Z}_{N^\gamma}$ but its roots are all in $\mathbb{Z}^*_N$). Then, for each $i \in \{0, \ldots, n\}$, set $U_i = (1+N)^{\alpha_i} a_i^{N^\gamma} \bmod N^{\gamma+1}$, where $(a_0, \ldots, a_n) \xleftarrow{\$} (\mathbb{Z}^*_N)^{n+1}$ and with $\alpha_n = 1$.
  3. Set the evaluation key as $s' = \{N, \vec{U}\}$, where $\vec{U}$ is the vector $\vec{U} = (U_0, \ldots, U_n)$, and the domain of the function as $\{0, \ldots, 2^{\gamma\lambda/2} - 1\}$. The trapdoor is defined to be $t' = \mathrm{lcm}(p-1, q-1)$.

- **Evaluation:** to evaluate $G_{\mathrm{abn}}(s', \theta, x)$, where $x \in \{0, \ldots, 2^{\gamma\lambda/2} - 1\}$ and $\theta \in \{0,1\}^\lambda$, compute $c = \left(\prod_{j=0}^n U_i^{(\theta^i \bmod N^\gamma)}\right)^x \bmod N^{\gamma+1}$.

- **Inversion:** for a branch θ, $c = G_{\mathrm{abn}}(s', \theta, x)$ is a Damgård-Jurik encryption of $y = P(\theta)x \bmod N^\gamma$. Using the trapdoor $t' = \mathrm{lcm}(p-1, q-1)$, the inversion procedure first applies the decryption algorithm of [DJ01] to obtain $y \in \mathbb{Z}_{N^\gamma}$ and returns $x = yP(\theta)^{-1} \bmod N^\gamma$.

As in [RS09, BFO08], $G_{\mathrm{abn}}(s', \theta, \cdot)$ has image size smaller than N in lossy mode. Hence, the average min-entropy of x can be shown to be at least $\tilde{H}_\infty(x | (G_{\mathrm{abn}}(s', \theta, x), N, \vec{U})) \geq \gamma\lambda/2 - \log(N)$ when $\theta \in I$.

We also note that the ABN function $G_{\mathrm{abn}}(s', \theta, \cdot)$ is not strictly injective for each branch $\theta \notin I$, but only for those such that $\gcd(P(\theta), N^\gamma) = 1$. However, the fraction of branches $\theta \in \{0,1\}^\lambda$ such that $\gcd(P(\theta), N^\gamma) \neq 1$ is bounded by $2/\min(p, q)$, which is negligible.

Moreover, the proof of theorem C.4.4 is not affected if the TBE scheme is instantiated with this particular ABN function and the LTDF of [RS09, BFO08]. As long as factoring is hard (which is implied by the Composite Residuosity assumption), the adversary has negligible chance of making decryption queries w.r.t. to such a problematic tag θ.

Lemma C.4.5 The above ABN function satisfies the hidden lossy set property under the Decisional Composite Residuosity assumption.

Proof: For the sake of contradiction, let us consider an adversary $\mathcal{A}$ that distinguishes two ABN functions with lossy sets $I_A = \{\theta_{A,1}^\star, \ldots, \theta_{A,n}^\star\}$ and $I_B = \{\theta_{B,1}^\star, \ldots, \theta_{B,n}^\star\}$ of its choice. Let $P_A[T]$ and $P_B[T]$ be the n^{th} degree polynomials having their roots in I_A and I_B, respectively. We consider a sequence of games starting with Game_A, where the adversary is given an ABN with lossy set I_A, and ending with Game_B where the ABN has lossy set I_B. Then, we consider a sequence of hybrid games where, for $j = 0, \ldots, n-1$, $\mathrm{Game}_{H,j}$ is defined to be a game where $U_0, \ldots, U_j$ are Damgård-Jurik encryptions of the coefficients of $P_A[T]$ until degree j whereas $U_{j+1}, \ldots, U_{n-1}$ encrypt the coefficients of $P_B[T]$. Obviously, any adversary distinguishing Game_A from $\mathrm{Game}_{H,0}$ implies a semantic security adversary against Damgård-Jurik and the same argument applies to subsequent game transitions. The result follows by noting that Game_B is identical to $\mathrm{Game}_{H,n-1}$. ∎

The above ABN function yields an IND-SO-CCA2 secure encryption scheme with ciphertexts of constant (*i.e.*, independent of n) size but a public key of size $O(n)$. Encryption and decryption

require $O(n)$ exponentiations as they entail an ABN evaluation. On the other hand, the private key has $O(1)$ size as well, which keeps the private storage very cheap. At the expense of sacrificing the short private key size, the decryption algorithm can be optimized by computing $x = G_{\text{abn}}^{-1}(t', \theta, c_2)$ (instead of $x = F_{\text{ltdf}}^{-1}(t, c_1)$) so as to avoid computing $G_{\text{abn}}(s', \theta, x)$ in the forward direction to check the validity of ciphertexts. In this case, the receiver has to store the coefficients $\alpha_0, \ldots, \alpha_{n-1}$ to evaluate $P(\theta)$ when inverting G_{abn}.

It is also possible to extend the DDH-based ABO function described in [PW08] into an ABN function. However, the next section describes a more efficient lossy TBE scheme based on the DDH assumption.

C.4.8 An IND-SO-stag-wCCA2 TBE Scheme from the DDH Assumption

The DDH problem informally consists in, given (g, g^x, g^y, g^z), to decide whether $z = xy$ or not (a rigorous definition is recalled in appendix

Rigorously,

Definition C.4.6 The **Decisional Diffie-Hellman** (DDH) problem in a group $\mathbb{G}$, is to distinguish the two distributions

$$\begin{aligned} D_1 &= \{x, y \xleftarrow{\$} \mathbb{Z}_p : (g, g^x, g^y, g^{xy})\}, \\ D_2 &= \{x, y \xleftarrow{\$} \mathbb{Z}_p; z \xleftarrow{\$} \mathbb{Z}_p \setminus \{xy\} : (g, g^x, g^y, g^z)\}. \end{aligned}$$

The **DDH assumption** posits that, for any PPT distinguisher $\mathcal{D}$, the following function is negligible

$$\begin{aligned} \mathbf{Adv}^{\text{DDH}}_{\mathbb{G},\mathcal{D}}(\lambda) = |\Pr[\mathcal{D}(\{(g, X, Y, Z) \xleftarrow{\$} D_1 : g, X, Y, Z\}) = 1] \\ - \Pr[\mathcal{D}(\{(g, X, Y, Z) \xleftarrow{\$} D_2 : g, X, Y, Z\}) = 1]|. \end{aligned}$$

The system builds on the DDH-based lossy encryption scheme of [NP01,PVW08,BHY09] and could be seen as a variant of the encryption scheme described in [CKS08, Section 6.2], which is itself situated half-way between the Cramer-Shoup [CS98, CS02] and CHK methodologies [CHK04].

Again, attention must be paid to the fact that the adversary sees $n > 1$ challenge ciphertexts with different tags. To apply the technique of [CKS08] (which uses ideas that were initially proposed for identity-based encryption [BB04a]) in the security proof, we need some function of the tag to cancel in the exponent for each target ciphertext. This issue can be addressed using the technique of [CS06].

$\mathsf{TBEKg}(1^\lambda)$**:** choose a group $\mathbb{G}$ of prime order $p > 2^\lambda$ with a generators $g, h \xleftarrow{\$} \mathbb{G}$. Pick $a_i, b_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 0, \ldots, n$, and compute $U_i = g^{a_i}$, $V_i = h^{a_i}$, $W_i = g^{b_i}$, $Z_i = h^{b_i}$ and $Y_1 = g^y$, $Y_2 = h^y$ for a random $y \xleftarrow{\$} \mathbb{Z}_p$. Set the public key as $pk = \{\mathbb{G}, g, h, \vec{U}, \vec{V}, \vec{W}, \vec{Z}, X_1, X_2\}$ and define the private key to be $sk = (\vec{a}, \vec{b}, y)$, for $(n+1)$-vectors $\vec{U} = (U_0, \ldots, U_n)$, $\vec{V} = (V_0, \ldots, V_n)$, $\vec{W} = (W_0, \ldots, W_n)$, $\vec{Z} = (Z_0, \ldots, Z_n)$, $\vec{a} = (a_0, \ldots, a_n)$ and $\vec{b} = (b_0, \ldots, b_n)$.

$\mathsf{TBEEnc}(pk, \theta, m)$**:** to encrypt m under the tag $\theta \in \mathbb{Z}_p$ given pk,

1. Choose $r, s \xleftarrow{\$} \mathbb{Z}_p$ and compute $C_0 = m \cdot Y_1^r \cdot Y_2^s$, $C_1 = g^r \cdot h^s$.
2. Set $C_2 = (\prod_{j=0}^n U_j^{\theta^j})^r \cdot (\prod_{j=0}^n V_j^{\theta^j})^s$ and $C_3 = (\prod_{j=0}^n W_j^{\theta^j})^r \cdot (\prod_{j=0}^n Z_j^{\theta^j})^s$.

Set the ciphertext as $C = \Big(C_0, C_1, C_2, C_3\Big)$.

$\mathsf{TBEDec}(sk, \theta, C)$**:** given $sk = (\vec{a}, \vec{b}, y)$, θ and $C = (C_0, C_1, C_2, C_3)$, return $\perp$ if $C_2 \neq C_1^{\sum_{j=0}^{n} a_j \theta^j}$ or $C_3 \neq C_1^{\sum_{j=0}^{n} b_j \theta^j}$. Otherwise, return $m = C_0/C_1^y$.

This scheme is separable since three functions f_1, f_2 and f_3 can be defined so that $C_0 = f_1(pk, m, (r, s))$, $C_1 = f_2(pk, (r, s))$ and $(C_2, C_3) = f_3(pk, \theta, (r, s))$. The chameleon-hash-based transformation thus applies and we only have to prove that the TBE system satisfies IND-SO-stag-wCCA2 security.

Theorem C.4.7 For any adversary $\mathcal{A}$ making q decryption queries, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\text{IND-SO-stag-wCCA2}}(\lambda) \leq \mathbf{Adv}_{\mathbb{G}}^{\text{DDH}}(\lambda) + q/2^{\lambda}.$$

Proof: The proof consists of a sequence of games, the first one of which is the real game. In all games, we call S_i the event that the adversary $\mathcal{A}$ outputs 1 in Game_i.

Game$_0$: the adversary chooses n tags $\theta_1^\star, \ldots, \theta_n^\star$ and is supplied with a public key for which $\vec{U}$, $\vec{V}$, $\vec{W}$, $\vec{Z}$, Y_1, Y_2 are generated such that $Y_1 = g^y$, $Y_2 = h^y$, for some $y \xleftarrow{\$} \mathbb{Z}_p$, and $U_i = g^{a_i}$, $V_i = h^{a_i}$, $W_i = g^{b_i}$ and $Z_i = h^{b_i}$ for $i \in \{0, \ldots, n\}$ where $(a_0, \ldots, a_n) \xleftarrow{\$} (\mathbb{Z}_p)^{n+1}$ and $(b_0, \ldots, b_n) \xleftarrow{\$} (\mathbb{Z}_p)^{n+1}$.
The adversary $\mathcal{A}$ makes decryption queries which the simulator $\mathcal{D}$ handles using $sk = (\vec{a}, \vec{b}, y)$, where $\vec{a} = (a_0, \ldots, a_n)$, $\vec{b} = (b_0, \ldots, b_n)$. After polynomially-many decryption queries, $\mathcal{A}$ makes a unique challenge query for a message distribution $\mathcal{M}$ of his choice. Then, $\mathcal{D}$ uniformly samples n plaintexts $(m_1^\star, \ldots, m_n^\star) \xleftarrow{\$} \mathcal{M}^n$ and generates a vector of ciphertexts $\mathbf{C}^\star = (\mathbf{C}[1]^\star, \ldots, \mathbf{C}[n]^\star)$. For $i \in \{1, \ldots, n\}$, let us call $r_i^\star, s_i^\star \in \mathbb{Z}_p$ the random exponents that are used to generate $\mathbf{C}[i]^\star$ such that $\mathbf{C}[i]^\star = (C_{i,0}^\star, C_{i,1}^\star, C_{i,2}^\star, C_{i,3}^\star)$ is equal to

$$\Big(m_i^\star \cdot Y_1^{r_i^\star} \cdot Y_2^{s_i^\star},\ g^{r_i^\star} \cdot h^{s_i^\star},\ (\prod_{j=0}^{n} U_j^{t^j})^{r_i^\star} \cdot (\prod_{j=0}^{n} V_j^{t^j})^{s_i^\star},\ (\prod_{j=0}^{n} W_j^{t^j})^{r_i^\star} \cdot (\prod_{j=0}^{n} Z_j^{t^j})^{s_i^\star}\Big).$$

After having obtained the vector $\mathbf{C}^\star$, $\mathcal{A}$ makes further decryption queries (C, θ) such that $\theta \notin \{\theta_1^\star, \ldots, \theta_n^\star\}$. At some point, he makes a corruption query and chooses a subset $I \subset \{1, \ldots, n\}$ such that $\#I = n/2$. At this stage, $\mathcal{D}$ returns $\{(m_i^\star, (r_i^\star, s_i^\star))\}_{i \in I}$. As for indices $i \in \{1, \ldots, n\} \setminus I$ corresponding to unopened plaintexts, $\mathcal{D}$ only returns the actual plaintexts $\{m_i^\star\}_{i \notin I}$. The adversary $\mathcal{A}$ makes further decryption queries (C, θ) subject to the rule that $\theta \notin \{\theta_1^\star, \ldots, \theta_n^\star\}$. We call S_0 the event that $\mathcal{A}$ eventually outputs 1.

Game$_1$: is the same as Game_0 but we modify the generation of the public key. Namely, to generate $pk = \{\mathbb{G}, g, h, f, \vec{U}, \vec{V}, \vec{W}, \vec{Z}, Y_1, Y_2\}$, the simulator $\mathcal{D}$ first computes $X_1 = g^x$ and $X_2 = h^x$, for a random $x \xleftarrow{\$} \mathbb{Z}_p$, and calculates Y_1, Y_2 and vectors $(\vec{U}, \vec{V}, \vec{W}, \vec{Z})$ in the following way. The simulator $\mathcal{D}$ uniformly picks $\alpha_n, \beta_0, \ldots, \beta_n, \gamma_0, \ldots, \gamma_n \xleftarrow{\$} \mathbb{Z}_p$. It obtains coefficients $\alpha_{n-1}, \ldots, \alpha_0$ by expanding the polynomial $P[T] = \alpha_n(T - \theta_1^\star) \ldots (T - \theta_n^\star) = \alpha_n T^n + \alpha_{n-1} T^{n-1} + \cdots + \alpha_1 T + \alpha_0$. Then, it defines $Y_1 = g^{\omega_1} X_1^{\omega_2}$ and $Y_2 = h^{\omega_1} X_2^{\omega_2}$ for randomly drawn $\omega_1, \omega_2 \xleftarrow{\$} \mathbb{Z}_p$. For each $i \in \{0, \ldots, n\}$, it sets

$$U_i = X_1^{\alpha_i} g^{\beta_i},\ V_i = X_2^{\alpha_i} h^{\beta_i},\ W_i = Y_1^{\alpha_i} g^{\gamma_i},\ Z_i = Y_2^{\alpha_i} h^{\gamma_i}.$$

This implicitly defines private keys elements $\vec{a}, \vec{b}$ and y to be $a_i = \alpha_i x + \beta_i$, $b_i = \alpha_i y + \gamma_i$, for $i \in \{0, \ldots, n\}$, and $y = \omega_1 + x\omega_2$. The distribution of pk is not modified and we have $\Pr[S_1] = \Pr[S_0]$.

Game$_2$: we now modify the decryption oracle. For a decryption query on (C, θ) where $C = (C_0, C_1, C_2, C_3)$ with $\theta \notin \{\theta_1^\star, \ldots, \theta_n^\star\}$, $\mathcal{D}$ evaluates the polynomials $Q_2[T] = \sum_{j=0}^{n} \beta_i T^j$ and $Q_3[T] = \sum_{j=0}^{n} \gamma_j T^j$ for $T = \theta$ and computes $A_i = (C_i / C_1^{Q_i(\theta)})^{1/P(\theta)}$ for $i \in \{2, 3\}$. The consistency of the ciphertext is verified by checking whether $C_1^{\omega_1} A_2^{\omega_2} = A_3$ and returning $\perp$ if this is not the case.
This consistency check stems from the "Twin Diffie-Hellman trapdoor test" [CKS08, Theorem 2], the idea of which is the following. If C is well-formed, for any pair (r, s) such that $C_1 = g^r h^s$, we must have $A_2 = X_1^r X_2^s$ and $A_3 = Y_1^r Y_2^s$ (so that $A_3 = C_1^{\omega_1} A_2^{\omega_2}$ and the test is successful).
Let us assume that there exists no r, s such that $C_1 = g^r h^s$, $C_2 = (g^{Q_2(\theta)} X_1^{P(\theta)})^r (h^{Q_2(\theta)} X_2^{P(\theta)})^s$ and $C_3 = (g^{Q_3(\theta)} Y_1^{P(\theta)})^r (h^{Q_3(\theta)} Y_2^{P(\theta)})^s$. The trapdoor test amounts to check whether there exists $\tau = r + \log_g(h) s$ such that $C_1 = g^\tau$, $C_2 = (g^{Q_2(\theta) + xP(\theta)})^\tau$ and $C_3 = (g^{Q_3(\theta) + yP(\theta)})^\tau$. If this is not the case, $\mathcal{D}$ obtains $A_2 = g^{x\tau_1}$ and $A_3 = g^{y\tau_2}$ such that either $\tau_1 \neq \tau$ or $\tau_2 \neq \tau$. It is easy to see that the trapdoor test cannot be satisfied if $\tau = \tau_1$ and $\tau \neq \tau_2$ and we thus assume that $\tau_1 \neq \tau$. In this case, we can write $A_2 = g^{x(\tau + \tau_1')}$, for some $\tau_1' \neq 0$, and the value $C_1^{\omega_1} A_2^{\omega_2}$ can in turn be written $g^{\tau(\omega_1 + x\omega_2)} \cdot g^{x\tau_1'\omega_2} = g^{\tau y} \cdot g^{x\tau_1'\omega_2}$, which is uniformly random from $\mathcal{A}$'s view (since the product $x\omega_2$ is perfectly hidden). Moreover, conditionally on a fixed $y = \log_g(Y_1)$, the distribution of A_3 does not depend on $x\omega_2$ since $A_3 = (C_3 / C_1^{Q_3(\theta)})^{1/P(\theta)}$ can be expressed as $A_3 = C_1^y \cdot (h^{\frac{Q_3(\theta)}{P(\theta)}} \cdot Y_2)^{s' - s}$ where (s, s') are such that $s' = s$ if $C_3 = C_1^{Q_3(\theta) + yP(\theta)}$. It comes that the condition $A_3 = C_1^{\omega_1} A_2^{\omega_2}$ cannot be satisfied with better probability than $1/q$ and C is thus rejected with probability $1 - 1/q$.
If the check succeeds, $\mathcal{D}$ returns $m = C_0 / A_3$. We have $|\Pr[S_2] - \Pr[S_1]| \leq q/p \leq q/2^\lambda$ as Game 2 and Game 1 are identical until $\mathcal{D}$ accepts a ciphertext that would have been rejected in Game 1.

Game$_3$: we modify again the generation of pk. Now, $\mathcal{D}$ computes $X_1 = g^x$ and $X_2 = h^{x'}$, where $x \xleftarrow{\$} \mathbb{Z}_p$, $x' \xleftarrow{\$} \mathbb{Z}_p \setminus \{x\}$ (instead of $X_2 = h^x$). All other calculations (including the generation of $\mathbf{C}^\star$ and the decryption oracle) remain unchanged. In particular, $\mathcal{D}$ still knows the encryption exponents $r_i^\star, s_i^\star \in \mathbb{Z}_p$ that are used to encrypt $\mathbf{C}[i]^\star$, for $i \in \{1, \ldots, n\}$, and the exponents $\vec{\alpha}, \vec{\beta}, \vec{\gamma}$ used in the previous game.
The decryption oracle still consistently handles decryption queries as they involve tags $\theta \notin \{\theta_1^\star, \ldots, \theta_n^\star\}$. For any queried ciphertext $C = (C_0, C_1, C_2, C_3)$, given that $\log_g(X_1) \neq \log_h(X_2)$, there always exist (r, s) such that $C_1 = g^r h^s$ and $C_2 = (g^{Q_2(\theta)} X_1^{P(\theta)})^r (h^{Q_2(\theta)} X_2^{P(\theta)})^s$. For these values (r, s), the decryption oracle obtains $A_2 = X_1^r X_2^s$. Likewise, there always exists a pair of integers (r', s') satisfying $C_1 = g^{r'} h^{s'}$ and $C_3 = (g^{Q_3(\theta)} Y_1^{P(\theta)})^{r'} (h^{Q_3(\theta)} Y_2^{P(\theta)})^{s'}$ and $\mathcal{D}$ obtains $A_3 = Y_1^{r'} Y_2^{s'}$. If C is well-formed, we have $(r, s) = (r', s')$ and the oracle returns $m = C_0 / A_3$ as in previous games. If $(r, s) \neq (r', s')$, A_3 can be written $A_3 = Y_1^r Y_2^{s_1}$, for some $s_1 \neq s$, so that $A_3 / (C_1^{\omega_1} A_2^{\omega_2}) = Y_2^{s_1 - s} \neq 1_{\mathbb{G}}$ and the test rejects C.

Any notable difference between Game$_3$ and Game$_2$ would give a DDH-adversary. To construct a distinguisher that bridges between these games, we consider a DDH instance $(g, h, X_1 = g^x, X_2)$ and generate the public key as in Game$_1$. It comes that key generation proceeds as in Game$_2$ if $X_2 = h^x$ and mirrors Game$_3$ otherwise. Hence, $|\Pr[S_3] - \Pr[S_2]| \leq \mathbf{Adv}_{\mathbb{G}}^{\mathrm{DDH}}(\lambda)$.
In Game$_3$, ciphertexts $\mathbf{C}[i]^\star$ are statistically independent of plaintexts. Indeed, they are of the

form

$$(C^\star_{i,0}, C^\star_{i,1}, C^\star_{i,2}, C^\star_{i,3}) = \Big(m^\star_i \cdot Y_1^{r^\star_i} Y_2^{s^\star_i},\ g^{r^\star_i} h^{s^\star_i},\ (g^{r^\star_i} h^{s^\star_i})^{Q_2(t^\star_i)},\ \ (g^{r^\star_i} h^{s^\star_i})^{Q_3(t^\star_i)}\Big),$$

so that, since $\mathcal{A}$ knows $Q_2(\theta^\star_i)$ and $Q_3(\theta^\star_i)$ in the information-theoretic sense, the information revealed by $C^\star_{i,1}$, $C^\star_{i,2}$, $C^\star_{i,3}$ is redundant and leaves p equally-likely candidates for the pair $(r^\star_i, s^\star_i)$. The value $Y_1^{r^\star_i} Y_2^{s^\star_i}$ is then easily seen to statistically hide $m^\star_i$ since $\log_g(Y_1) \neq \log_h(Y_2)$. Even an all-powerful adversary would be unable to tell whether he obtains the real plaintext $m^\star_i$ or a resampled one. The proof is completed using a sequence of n hybrid games exactly as in the end of the proof of theorem C.4.4. ▌

As in the Paillier-based scheme, the number n of target ciphertexts must be known at key generation since public keys have size $O(n)$. As long as n is not too large, the encryption cost remains acceptable: if n is a linear polynomial in λ for instance, the encryption algorithm has complexity $O(\lambda^4)$. Hofheinz recently showed [Hof12] how to avoid this annoying linear dependency. He notably described a Paillier-based trapdoor functions where new lossy branches can be created at will and with a constant-size public key.

On the other hand, ciphertexts consist of a constant number of group elements and decryption entails a constant number of exponentiations.

C.5 Conclusion

We showed that lossy encryption, which is known to provide IND-SO-CPA secure encryption schemes, is implied by the re-randomizable encryption primitive as well as by $\binom{2}{1}$-Oblivious Transfer (and thus also by PIR, homomorphic encryption and smooth hash proof systems).

Our constructions explain an existing scheme and give rise to new IND-SO-CPA secure cryptosystems based on the Decisional Composite Residuosity (DCR) and Quadratic Residuosity (QR) assumptions. These new schemes retain the efficiency of underlying protocols and immediately yield simple and efficient IND-SO-COM secure commitments. From Paillier's cryptosystem, we additionally obtained the most bandwidth-efficient SEM-SO-CPA secure encryption scheme to date and the first one based on the DCR assumption.

In the chosen-ciphertext selective opening scenario, we described new schemes fitting indistinguishability and simulation-based definitions. As for the former, we showed how to reach security in its sense using schemes with short ciphertexts. The recent results of Hofheinz [Hof12] show how create All-But-Many Lossy Functions, which can be used to eliminate the $\mathcal{O}(n)$ complexity in terms of public key size in our constructions while retaining short ciphertexts. This significantly increases the utility of our constructions.

Acknowledgements

We thank Yuval Ishai for suggesting a connection between Oblivious Transfer and Lossy Encryption.

Brett Hemenway was supported in part by NSF VIGRE Fellowship and NSF grants 0716835, 0716389, 0830803 and 0916574. Rafail Ostrovsky's research is supported in part by NSF grants 0830803, 09165174, 106527 and 1118126, US-Israel BSF grant 2008411, grants from OKAWA Foundation, IBM, Lockheed-Martin Corporation and the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views

expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Benoît Libert acknowledges the Belgian Fund for Scientific Research (F.R.S.-F.N.R.S.) for his "Collaborateur scientifique" fellowship and the BCRYPT Interuniversity Attraction Pole. Damien Vergnaud was supported in part by the European Commission through the ICT Program under contract ICT-2007-216676 ECRYPT II.

C.A Selective Opening Secure Commitments

C.A.1 Re-Randomizable One-Way Functions

A family of functions $\mathcal{F}$, indexed by a security parameter λ is called a *re-randomizable one-way function* family if the following conditions are satisfied

- **Efficiently Computable:** For all $f \in \mathcal{F}$, the function $f : M \times R \to Y$ is efficiently computable.
- **One-Way:** For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

  $$\Pr\left[f \leftarrow \mathcal{F}; (m_0, m_1, st) \leftarrow \mathcal{A}_1(f); b \leftarrow \{0,1\}; r \leftarrow R; b' \leftarrow \mathcal{A}_2(f(m_b, r), st) : b = b'\right] < \frac{1}{2} + \nu$$

  for some negligible function ν (of λ).
- **Injective on the first input:** For all $m \neq m' \in M$, and $r, r' \in R$, $f(m, r) \neq f(m', r')$. This is equivalent to the statement $f(m, R) \cap f(m', R) = \emptyset$ for all $m \neq m' \in M$.
- **Re-randomizable:** For each f, there exists and efficient function ReRand such that, for all $m \in M$ and $r_0 \in R$, we have

  $$\{r \leftarrow R; f(m, r)\} \approx_s \{r \leftarrow \mathsf{coins}(\mathsf{ReRand});\ \mathsf{ReRand}(f(m, r_0), r)\}.$$

It is easy to see that the encryption algorithm from a re-randomizable encryption scheme is immediately a re-randomizable one-way function. We note, however, that re-randomizable one-way functions are a significantly weaker primitive since we do not require any kind of trapdoor.

C.A.2 Commitments from Re-Randomizable One-Way Functions

We begin by describing a construction of a simple bit commitment scheme that arises from any re-randomizable one-way function. Let $\mathcal{F}$ be a re-randomizable one-way function family. The bit commitment system is depicted on figure C.4.

Parameter Generation:	**Commitment:**
$(f, \mathsf{ReRand}) \leftarrow \mathcal{F}(1^\lambda)$	$r' \leftarrow \mathsf{coins}(\mathsf{ReRand})$
$r_0, r_1 \leftarrow R$	$\mathrm{Com}(b, r') = \mathsf{ReRand}(c_b, r')$
$c_0 = f(b_0, r_0)$	**De-commitment:**
$c_1 = f(b_1, r_1)$	To de-commit, simply reveal the randomness r'.

Figure C.4: Commitments from re-randomizable one-way functions

This scheme has a number of useful properties. If $b_0 = b_1$, the scheme is statistically hiding by the properties of ReRand. Alternatively, if $b_0 \neq b_1$, the scheme is perfectly binding by

the injectivity of f on its first input. Now, the two modes are indistinguishable by the one-wayness of f. Combining this with the preceding observations, we also obtain that the scheme is computationally binding if $b_0 = b_1$ and computationally hiding if $b_0 \neq b_1$.

The security analysis is very straightforward but, as this will be the foundation of all our constructions, we include it hereafter.

Lemma C.A.1 If $b_0 = b_1$, the commitment scheme of figure C.4 is statistically hiding. If $b_0 \neq b_1$, then it is perfectly binding.

Proof: If $b_0 = b_1$, we have

$$\{r' \leftarrow \mathsf{coins}(\mathrm{Com}) : \mathrm{Com}(0, r')\} \approx_s \{s' \leftarrow \mathsf{coins}(\mathrm{Com}) : \mathrm{Com}(1, s')\},$$

by the definition of ReRand. On the other hand, if $b_0 \neq b_1$, $\mathrm{Com}(0, r) \in f(b_0, R)$ and $\mathrm{Com}(1, s) \in f(b_1, R)$, but by the injectivity on the first input, these sets are necessarily disjoint . ∎

Lemma C.A.2 Instantiations of the scheme with $b_0 = b_1$ and $b_0 \neq b_1$ are computationally indistinguishable.

Proof: This is exactly the one-way property of f. ∎

Corollary C.A.3 If $b_0 = b_1$, the scheme is computationally binding. If $b_0 \neq b_1$, it is computationally hiding.

Proof: Since the scheme is perfectly binding when $b_0 \neq b_1$, breaking the binding property amounts to a proof that $b_0 = b_1$. Since the two modes are computationally indistinguishable, no computationally bounded adversary can create such a "proof." Similarly, since the scheme is perfectly hiding when $b_0 = b_1$, breaking the hiding property amounts to showing that $b_0 \neq b_1$, since the two modes are computationally indistinguishable, no probabilistic polynomial-time adversary can break the hiding property. ∎

The ability to choose whether the commitment scheme will be statistically hiding or perfectly binding is a valuable property, but it is the fact that this choice can be hidden *from the committer* that makes this construction truly useful.

C.A.3 Definitions of Selective Opening Secure Commitments

Definition C.A.4 (Indistinguishability of commitments under selective openings). A non-interactive commitment scheme $(\mathrm{Com}, \mathrm{Dec})$ is indistinguishable under selective openings (or IND-SO-COM secure) if, for any polynomial n, any n-message distribution $\mathcal{M}$ supporting efficient conditional resampling and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have

$$\left|\Pr\left[\mathcal{A}^{\mathsf{ind\text{-}so\text{-}real}} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{ind\text{-}so\text{-}ideal}} = 1\right]\right| < \nu$$

for some negligible function ν, and where the games ind-so-real and ind-so-ideal are defined as follows

More explicitly, in the real game, the challenger samples messages $\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$ from the joint message distribution and picks random coins $r_1, \ldots, r_n \leftarrow \mathsf{coins}(\mathrm{Com})$ to compute n commitments $\mathrm{Com}(m_1, r_1), \ldots, \mathrm{Com}(m_n, r_n)$ which are sent to $\mathcal{A}$ along with a description

IND-SO-COM (Real):	**IND-SO-COM (Ideal):**
$\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$	$\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$
$r_1, \ldots, r_n \leftarrow \mathsf{coins}(\mathrm{Com})$	$r_1, \ldots, r_n \leftarrow \mathsf{coins}(\mathrm{Com})$
(I, st)	(I, st)
$\quad \leftarrow \mathcal{A}_1\big(\mathsf{par}, \mathrm{Com}(m_1, r_1), \ldots, \mathrm{Com}(m_n, r_n)\big)$	$\quad \leftarrow \mathcal{A}_1\big(\mathsf{par}, \mathrm{Com}(m_1, r_1), \ldots, \mathrm{Com}(m_n, r_n)\big)$
$b \leftarrow \mathcal{A}_2\big(st, \mathrm{Dec}(\mathrm{Com}(m_i, r_1))_{i \in I}, \mathbf{m}\big)$	$\mathbf{m}' = (m'_1, \ldots, m'_n) \leftarrow \mathcal{M}_{\lvert I, \mathbf{m}[I]}$
	$b \leftarrow \mathcal{A}_2\big(st, \mathrm{Dec}(\mathrm{Com}(m_i, r_i))_{i \in I}, \mathbf{m}'\big)$

Figure C.5: IND-SO-COM Security

of public parameters par. The adversary $\mathcal{A}$ responds by choosing a subset $I \subset \{1, \ldots, n\}$ of size $n/2$. Then, the challenger de-commits $\{\mathrm{Com}(m_i, r_i)\}_{i \in I}$ and hands the result $\{(m_i, r_i)\}_{i \in I}$ to $\mathcal{A}$. Finally, the challenger sends $\mathbf{m}$ to the adversary $\mathcal{A}$ who eventually outputs a bit $b \in \{0, 1\}$.

The ideal game proceeds identically to the real game until the opening query. At this stage, the challenger still de-commits $\{\mathrm{Com}(m_i, r_i)\}_{i \in I}$ by revealing $\{(m_i, r_i)\}_{i \in I}$ to $\mathcal{A}$. Instead of revealing $\mathbf{m}$ however, it samples a new vector $\mathbf{m}' \leftarrow \mathcal{M}_{|I, \mathbf{m}[I]}$ from $\mathcal{M}$ conditioned on the fact that $m_i = m'_i$ for $i \in I$ and sends it to $\mathcal{A}$ who eventually outputs a bit $b \in \{0, 1\}$.

C.A.4 IND-SO-COM Constructions from Re-Randomizable One-Way Functions

To construct an IND-SO-COM secure commitment scheme, it suffices to create a statistically hiding commitment scheme as was demonstrated by Bellare, Hofheinz and Yilek [BHY09].

Theorem C.A.5 [BHY09] Statistically-hiding commitment schemes are IND-SO-COM secure.

Since the commitment scheme constructed in Appendix C.A.2 is statistically hiding when $b_0 = b_1$, we obtain the following corollary

Corollary C.A.6 Re-randomizable one-way functions imply non-interactive IND-SO-COM commitments.

Since re-randomizable encryptions imply re-randomizable one-way functions, we have

Corollary C.A.7 Re-randomizable encryption implies non-interactive IND-SO-COM secure commitments.

Perhaps more interesting is the case when $b_0 \neq b_1$. The commitment scheme constructed in Appendix C.A.2 is no longer perfectly hiding, so that Theorem C.A.5 doesn't apply. In this case, we can still achieve IND-SO-COM security by using the indistinguishability of the two modes. Roughly, this follows because an IND-SO-COM adversary must have similar probabilities of success against both modes, otherwise it could be used to distinguish the modes. We then obtain the following Corollary.

Corollary C.A.8 Re-randomizable one-way functions imply perfectly-binding IND-SO-COM commitments.

Since re-randomizable encryptions imply re-randomizable one-way functions, we have

Corollary C.A.9 Re-randomizable encryption implies perfectly binding non-interactive IND-SO-COM secure commitments.

Proof: The proof uses an equivalent definition of IND-SO-COM security where the adversary $\mathcal{A}$ is presented with a challenger that either plays the real game or the ideal one depending on the value of a secret bit, which $\mathcal{A}$ aims to guess.
Towards a contradiction, suppose there exists an IND-SO-COM adversary $\mathcal{A}$ that succeeds against the protocol with probability $\frac{1}{2} + \epsilon$ when $b_0 = b_1$. We will use $\mathcal{A}$ to construct a distinguisher D for the one-way game against the underlying re-randomizable one-way function f. In the one-wayness game against f, the challenger samples a function f and sends it to D. D will respond by sending $\{0, 1\}$ to the one-wayness challenger and the latter samples $r \leftarrow R$ and sends $e = f(b, r)$ to D. Now, D samples $r' \leftarrow R$ and generates $e' = f(0, r')$. Then, D instantiates the commitment protocol by setting $c_0 = e, c_1 = e'$ and plays the IND-SO-COM game with the adversary $\mathcal{A}$. If $\mathcal{A}$ wins, D guesses $b = 1$ whereas, if $\mathcal{A}$ loses, D bets that $b = 0$. From Theorem C.A.5, we know that, if $b = 0$, then $\mathcal{A}$ succeeds with advantage ν for some negligible function ν. On the other hand, by hypothesis, if $b = 1$, $\mathcal{A}$ wins the IND-SO-COM game with advantage ϵ. Now, it comes that

$$\begin{aligned}
\Pr[D \text{ wins }] &= \Pr[b = 1 \cap \ \mathcal{A} \text{ wins }] + \Pr[b = 0 \cap \ \mathcal{A} \text{ loses }] \\
&= \Pr[\mathcal{A} \text{ wins} | b = 1] \Pr[b = 1] + \Pr[\mathcal{A} \text{ loses} | b = 0] \Pr[b = 0] \\
&= \frac{1}{2}\left(\frac{1}{2} + \epsilon + \frac{1}{2} - \nu\right) = \frac{1}{2} + \frac{\epsilon - \nu}{2}.
\end{aligned}$$

Since ϵ is non-negligible and ν is negligible, D breaks the one-way property of f. ∎

We note that these constructions require trusted setup, which is necessary given the results of [BHY09], which showed a black-box separation between any primitive with a game-based definition of security and perfectly binding IND-SO-COM secure commitments without trusted setup.

C.B Homomorphic Encryption

A public key cryptosystem given by algorithms (G, E, D) is called *homomorphic* if

- The plaintext space forms a group X, with group operation $+$.
- The ciphertexts are members of a group Y.
- For all $x_0, x_1 \in X$, and for all $r_0, r_1 \in \mathsf{coins}(E)$, there exists an $r^* \in \mathsf{coins}(E)$ such that

$$E(pk, x_0 + x_1, r^*) = E(pk, x_0, r_0)E(pk, x_1, r_1).$$

Notice that we do not assume that the encryption is also homomorphic over the randomness, as is the case of most homomorphic encryption schemes, e.g. Elgamal, Paillier, and Goldwasser-Micali. We also do not assume that the image $E(pk, X, R)$ is the whole group Y, only that $E(pk, X, R) \subset Y$. Since the homomorphic property implies closure, we have that $E(pk, X, R)$ is a semi-group. Notice also, that while it is common to use the word "homomorphic" to describe the cryptosystem, encryption is *not* a homomorphism in the mathematical sense (although decryption is).

We now show some basic properties from all homomorphic encryption schemes. These facts are commonly used but, since our definition is weaker than the (implicit) definitions of homomorphic encryption that appear in the literature, it is important to note that they hold under this definition as well.

- $E(pk, X, R)$ is a group.
- $E(pk, 0, R)$ is a subgroup of $E(pk, X, R)$.
- For all $x \in X$, $E(pk, x, R)$ is the coset $E(pk, x, r)E(pk, 0, R)$.
- For all $x_0, x_1 \in X$, $|E(pk, x_0, R)| = |E(pk, x_1, R)|$.
- If y is chosen uniformly from $E(pk, 0, R)$, then $yE(pk, x, r)$ is uniform in $E(pk, x, R)$.
- $E(pk, X, R)$ is such that $E(pk, X, R) \simeq X \times E(pk, 0, R)$ and decryption is the homomorphism
$$E(pk, X, R) \to E(pk, X, R)/E(pk, 0, R) \simeq X.$$

We call a public key cryptosystem a *homomorphic public key encryption scheme*, if it is IND-CPA secure and homomorphic.

If we make the additional assumption that we can sample in a manner statistically close to uniform in the subgroup $E(pk, 0, R)$, then the homomorphic cryptosystem (G, E, D) will be re-randomizable.

Definition C.B.1 A homomorphic encryption scheme is said *uniformly sampleable* if there is a PPT algorithm sample such that the output of $\mathsf{sample}(pk)$ is statistically close to uniform on the group $E(pk, 0, R)$.

We note that, for all known homomorphic cryptosystems, we may define

$$\mathsf{sample}(pk) = \{r \leftarrow \mathsf{coins}(E) : E(pk, 0, r)\}.$$

It is not hard to see that this property *does not* automatically follow from the definition of homomorphic encryption. Since all known homomorphic schemes satisfy it however, they are re-randomizable.

C.B.1 Efficient Re-Randomizable Encryption from Uniformly Sampleable Homomorphic Encryption

Parameter Generation:	**Encryption:**
$(pk, sk) \leftarrow G(1^\lambda)$	$r' \leftarrow \mathsf{coins}(\mathsf{sample})$
$r \leftarrow \mathsf{coins}(E)$	$c' \leftarrow \mathsf{sample}(pk, r')$
$c = E(pk, b, r)$	return $c^a \cdot c'$
The public parameters are (pk, c)	**Decryption:**
	To decrypt a ciphertext c, simply return $D(c)$.

Figure C.6: Lossy Encryption from uniformly sampleable homomorphic encryption

The scheme of section C.3.1 only allows encrypting single bits. If the underlying cryptosystem (G, E, D) can encrypt more than one bit at a time, we can increase the efficiency of this system, by simply putting $c_0, c_1, \ldots, c_n$ into the public key, and an encryption of i will be $\mathsf{ReRand}(pk, c_i, r)$. In most cases, however, we can increase the size of encrypted messages without lengthening the public-key.

In particular, if $(G, E, D, \mathsf{sample})$ is a uniformly sampleable homomorphic encryption scheme and $\mathbb{Z}_N \hookrightarrow X$. Then, we can encrypt elements of $\{0, 1, \ldots, N-1\}$ instead of $\{0, 1\}$ as showed

by figure C.6.

If $c = E(pk, 0, r)$, the scheme is lossy since all encryptions will be uniformly distributed in the subgroup $E(pk, 0, R)$. In contrast, if $c = E(pk, 1, r)$, the scheme is injective by the correctness of the decryption algorithm. This is the natural construction when working with the Paillier or Damgård-Jurik cryptosystems. We must use caution when applying this construction to Elgamal since the inverse map $\mathbb{Z}_N \hookrightarrow X$ is not efficiently computable (it is the discrete log). In the context of commitments, it will not be a problem. On the other hand, when we want to view this as an encryption scheme for multi-bit messages, the lack of efficient inversion is an issue. Fortunately, a simple variant of Elgamal [NP01, PVW08, BHY09] is known to provide lossy encryptions from the DDH assumption. It is noteworthy that the "plain" Elgamal is itself re-randomizable although it is slightly less efficient than this modification.

C.C Simulation-Based Security

While we have mostly focused on an indistinguishability-based notion of security so far, Bellare *et al.* [BHY09] also formalized a simulation-based notion of security under selective openings. Their simulation-based definition of security intuitively seems stronger than the indistinguishability-based definition even though it still remains unknown whether SEM-SO-ENC implies IND-SO-ENC.

Definition C.C.1 (Semantic Security under selective openings). A public key cryptosystem (G, E, D) is *simulatable under selective openings* (SEM-SO-ENC secure) if, for any PPT n-message sampler $\mathcal{M}$, any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and any poly-time computable relation $\mathcal{R}$, there is an efficient simulator $S = (S_1, S_2)$ s.t.

$$\left|\Pr\left[\mathcal{A}^{\mathsf{sem\text{-}so\text{-}real}} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{sem\text{-}so\text{-}ideal}} = 1\right]\right| < \nu$$

for some negligible function ν, and where the games sem-so-real and sem-so-ideal are defined as follows

SEM-SO-ENC (Real):	**SEM-SO-ENC (Ideal):**
$\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$	$\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$
$r_1, \ldots, r_n \leftarrow \mathsf{coins}(E)$	$(I, st) \leftarrow S_1(1^\lambda)$
$(I, st) \leftarrow \mathcal{A}_1\big(pk, E(m_1, r_i), \ldots, E(m_n, r_n)\big)$	$w \leftarrow S_2\big(st, \{m_i\}_{i \in I}\big)$
$w \leftarrow \mathcal{A}_2\big(st, (m_i, r_i)_{i \in I}\big)$	Output $\mathcal{R}(\mathbf{m}, w)$
Output $\mathcal{R}(\mathbf{m}, w)$	

Figure C.7: SEM-SO-ENC Security

In the real game, the challenger samples $\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$ from the joint message distribution and picks random coins $r_1, \ldots, r_n \leftarrow \mathsf{coins}(E)$ to compute $E(m_1, r_1), \ldots, E(m_n, r_n)$ which are given to the adversary $\mathcal{A}$. The latter responds by choosing a $n/2$-subset $I \subset \{1, \ldots, n\}$ and gets back $\{(m_i, r_i)\}_{i \in I}$. The game ends with $\mathcal{A}$ outputting a string w and the value of the game is defined to be $\mathcal{R}(\mathbf{m}, w)$.

In the ideal game, the challenger samples messages $\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$ from the joint message distribution. Without seeing any encryptions, the simulator chooses a subset I and some state information st. After having seen the messages $\{m_i\}_{i \in I}$ and the state information but without seeing any randomness, the simulator outputs a string w. The result of the game

is $\mathcal{R}(\mathbf{m}, w)$.

In essence, simulation-based security demands that an efficient simulator be able to perform about as well as the adversary without having seen the challenge ciphertexts, the random coins or the public key.

In [BHY09], Bellare, Hofheinz and Yilek proved that any lossy encryption scheme endowed with an *efficient* opener procedure on lossy keys *is* SEM-SO-ENC secure.

Definition C.C.2 A *lossy public-key encryption scheme with efficient opening* is a tuple $(G_{\mathsf{inj}}, G_{\mathsf{lossy}}, E, D)$ satisfying Definition C.2.2, with the additional property that the algorithm opener is efficient, i.e.

- *Openability.* There is an *efficient* algorithm opener such that, if $(pk_{\mathsf{lossy}}, sk_{\mathsf{lossy}}) \leftarrow G_{\mathsf{lossy}}$, for all plaintexts $x_0, x_1 \in X$ and all $r \in \mathsf{coins}(E)$, with all but negligible probability, it holds that $E(pk_{\mathsf{lossy}}, x_0, r) = E(pk_{\mathsf{lossy}}, x_1, r')$, where $r' \leftarrow \mathsf{opener}(pk_{\mathsf{lossy}}, x_1, E(pk_{\mathsf{lossy}}, x_0, r))$.

Theorem C.C.3 [BHY09] Lossy Encryption with efficient opening is SEM-SO-ENC secure.

Proof: This is Theorem 2 in [BHY09].
The proof is straightforward, and we only sketch it here.

We proceed in a series of games.

- Game_0 is the real SEM-SO-ENC experiment.
- Game_1 is the same as Game_0 but the adversary is given a lossy public key instead of a real one.
- Game_2 instead of giving the adversary the real randomness $\{r_i\}_{i \in I}$, the challenger uses the efficient opener procedure to generate valid randomness.
- Game_3 instead of giving the adversary encryptions of m_i, the adversary is given encryptions of a dummy message ξ, but the adversary is still given openings to actual messages $\{m_i\}_{i \in I}$ obtained from the opener procedure.

Now, the simulator can simulate Game_3 with the adversary. The simulator generates a lossy key pair, and encrypts a sequence of dummy messages and forwards the encryptions to $\mathcal{A}$. The adversary, $\mathcal{A}$, replies with a set I, which S forwards to the challenger. Then S uses the efficient opener procedure to open the selected messages for $\mathcal{A}$. At which point $\mathcal{A}$ outputs a string w, and S outputs the same string. Since the outputs of $\mathcal{A}$ in Game_0 and Game_3 are computationally close, the outputs of S, and $\mathcal{A}$ in the real and ideal experiments will also be computationally close. ▌

C.C.1 Selective Opening Security from the Composite Residuosity Assumption

Here, we discuss the application of construction of section C.B.1 to Paillier's cryptosystem (a review of the details of the Paillier cryptosystem can be found in Appendix C.F).

By defining $\mathsf{ReRand}(c, r) = c \cdot E(pk, 0, r) \mod N^2$, we easily obtain a bandwidth-efficient IND-SO-ENC secure encryption scheme via our general construction in section C.B.1. It was already known how to obtain IND-SO-ENC security from the DCR assumption since Rosen and

Segev [RS09] and Boldyreva, Fehr and O'Neill [BFO08] showed how to build lossy-trapdoor functions using Composite Residuosity and lossy TDFs imply IND-SO secure encryption [BHY09]. By applying our construction to Paillier, we obtain a simpler and significantly more efficient construction than those following from [BFO08, RS09] under the same assumption.

While the results of [BHY09] imply that IND-SO-ENC secure encryptions follow from DCR, the question of SEM-SO-ENC secure encryptions was left open. The only previous construction of SEM-SO-ENC secure encryption was given in [BHY09] under the Quadratic Residuosity assumption (QR). From the Paillier and Damgård-Jurik cryptosystems, we readily obtain a lossy encryption scheme where the function opener is efficient. The results of [BY09, BHY09] then imply that the resulting encryption scheme achieves SEM-SO-ENC security.

To see that Paillier allows for efficient opening, recall that $E(pk, m, r) = g^m r^N \mod N^2$, where, in lossy mode, g is an N^{th} power (in which case, all ciphertexts are encryptions of 0) whereas its order is a multiple of N in injective mode. Then, any lossy ciphertext $c = E(pk, m, r)$ can be expressed as $c = r_1^N \mod N^2$ for some $r_1 \in \mathbb{Z}_N$, which the opener can compute as $r_1 = (c \mod N)^{1/N} \mod N$ (recall that $\gcd(N, \phi(N)) = 1$) using the factorization of N and $d = N^{-1} \mod \phi(N)$. Since g is itself a N^{th} residue in $\mathbb{Z}_{N^2}$, it can compute $g_0 \in \mathbb{Z}_N$ such that $g = g_0^N \mod N^2$ in the same way. To open c to $m \in \mathbb{Z}_N$, it has to find $r' \in \mathbb{Z}_N^*$ such that $r_1^N = g_0^{mN} r'^N \mod N^2$, which is easily obtained as $r' = r_1 g_0^{-m} \mod N$.

So, the efficiency of opener reduces to the efficiency of taking N^{th} roots modulo N, which is efficiently feasible

if the factorization of N is known. Hence, we immediately obtain a simple and efficient SEM-SO-ENC secure encryption system from the DCR assumption. We note that the possible use of Paillier as a lossy encryption scheme was implicitly mentioned in [YY05] but, to the best of our knowledge, its efficient openability property was never reported so far.

Corollary C.C.4 Under the DCR assumption, Paillier's cryptosystem is SEM-SO-ENC secure.

Since Paillier's cryptosystem (in the same way as the Damgård-Jurik extension) has smaller ciphertext expansion than the Goldwasser-Micali cryptosystem, we end up with a more efficient system than the only currently known SEM-SO-ENC secure cryptosystem.

C.D Lossy Encryption from Smooth Universal Hash Proof Systems

We recall the notion of a *smooth projective hash family* [CS02]. Let H be a hash family with keys in the set K, *i.e.* for each $k \in K$, $H_k : X \to \Pi$. Let $L \subset X$ and $\alpha : K \to S$. We require efficient evaluation algorithms such that, for any $x \in X$, $H_k(x)$ is efficiently computable using $k \in K$. Additionally, if $x \in L$ and a witness w for $x \in L$ is known, then $H_k(x)$ is efficiently computable given $x, w, \alpha(k)$.

Definition C.D.1 The set $(H, K, X, L, \Pi, S, \alpha)$ is a projective hash family if, for all $k \in K$, the action of H_k on the subset L is completely determined by $\alpha(k)$.

While $\alpha(k)$ determines the output of H_k on L, we need to ensure that it does not encode "too much" information on k. This is captured by the following definition of *smooth* projective hash family.

Definition C.D.2 Let $(H, K, X, L, \Pi, S, \alpha)$ be a projective hash family, and define two distributions Z_1, Z_2 taking values on the set $X \setminus L \times S \times \Pi$. For Z_1, we sample $k \xleftarrow{\$} K$, $x \xleftarrow{\$} X \setminus L$, and

set $s = \alpha(k)$, $\pi = H_k(x)$, for Z_2 we sample $k \xleftarrow{\$} K$, $x \xleftarrow{\$} X \setminus L$, and $\pi \xleftarrow{\$} \Pi$, and set $s = \alpha(k)$. The projective hash family is called *ν-smooth* if $\Delta(Z_1, Z_2) < \nu$.

The above basically says that, given $\alpha(k)$ and $x \in X \setminus L$, $H_k(x)$ is statistically close to uniform on Π.

Let $(H, K, X, L, \Pi, S, \alpha)$ be an ν-smooth projective hash family for some negligible function ν. We show a natural construction of Lossy Encryption. While smooth hash proof systems have a natural lossiness property, the constructions of IND-CPA secure encryption from [CS02] are not lossy encryption systems. The schemes described by Cramer and Shoup have two indistinguishable types of ciphertexts: "good" ciphertexts are generated in L while "bad" ciphertexts are sampled from $X \setminus L$. By turning their construction around, we can use their ciphertexts (in the IND-CCA1 version of their schemes) as public keys and their public keys as our ciphertexts to get a construction of Lossy Encryption.

- **Injective key generation:** Sample an element $x \in L$, along with the corresponding witness w.
  Set $PK = x$, $SK = w$.
- **Lossy key generation:** Sample an $x \in X \setminus L$. Set $PK = x$, $SK = \perp$.
- **Encryption:** To encrypt a message $m \in \Pi$, pick $k \xleftarrow{\$} K$, and output $c = (\alpha(k), H_k(x) + m)$, where $H_k(x)$ is efficiently computable without the witness w because k is known.
- **Decryption:** Given a ciphertext $c = (\alpha(k), \pi)$, use the witness w and $\alpha(k)$ to compute $H_k(x)$. Output $m = \pi - H_k(x)$.

The correctness of decryption follows immediately from the definitions and the indistinguishability of modes follows immediately from the hardness of the subset decision problem $L \subset X$. It only remains to see that, in lossy mode, the ciphertext is statistically independent of the plaintext m. But this follows immediately from the ν-smoothness of the hash proof system. Thus we arrive at

Lemma C.D.3 The scheme outlined above is a Lossy Encryption scheme.

The DDH-based lossy cryptosystem of [KN08,BY09,BHY09] is easily seen to be a particular case of this construction. Given public parameters $(g, h) \in \mathbb{G}$ for a group $\mathbb{G}$ of prime order p, we define $X = \mathbb{G}^2$ and L as the language $L = \{(Y_1, Y_2) = (g^y, h^y) : y \in \mathbb{Z}_p\}$, so that $w = y$ serves as a witness for the membership in L. We also define k to be a random pair $(r, s) \in (\mathbb{Z}_p)^2$ and $\alpha(k) = g^r \cdot h^s$ in such a way that $H_k((Y_1, Y_2)) = Y_1^r \cdot Y_2^s$ is easily computable using (r, s) and independent of $\alpha(k)$ when $(Y_1, Y_2) \notin L$.

Other known projective hash functions (e.g., [CS02]) immediately suggest new lossy encryption systems based on the Composite and Quadratic Residuosity assumptions that differ from currently known schemes. Yet another realization can be readily obtained from the Decision Linear assumption [BBS04], which is believed to be weaker than DDH.

C.E Chosen-Ciphertext Security: Simulatability

The simulation-based definition of [BY09,BHY09] also extends to the chosen-ciphertext scenario and involves an efficiently computable relation $\mathcal{R}$.

- **Selective opening query:** let $\mathcal{M}$ be a message distribution. The challenger samples a n-vector $\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$ and generates

$$(c_1, \ldots, c_n) = (E(pk, m_1, r_1), \ldots, E(pk, m_n, r_n)),$$

  which are sent the adversary. We call $c_1, \ldots, c_n$ the *target ciphertexts.*

- **Corruption query:** the adversary chooses a subset $I \subset [n]$ of cardinality $\#I = n/2$ and sends I to the challenger. The challenger then sends $\{(m_i, r_i)\}_{i \in I}$ to the Adversary.

  The challenger then sends $\{m_j\}_{j \notin I}$ to the adversary.

- **Decryption queries:** the adversary $\mathcal{A}$ chooses a ciphertext c that has never appeared as a target ciphertext, and sends c to the challenger. If c is a valid ciphertext (*i.e.*, $D(c) \neq \perp$) then the challenger responds with $m = D(c)$.

After adaptively making polynomially many queries, with at most one of them being a selective opening query, the adversary outputs w, and the value of the game is $\mathcal{R}(\mathbf{m}, w)$.

In the ideal game, the challenger samples $\mathbf{m} = (m_1, \ldots, m_n) \leftarrow \mathcal{M}$.

- The simulator chooses a subset, $I \leftarrow S_1$.
- The simulator views the chosen messages and outputs a w, $w \leftarrow S_2(\{m_i\}_{i \in I})$.

The value of the game is $\mathcal{R}(\mathbf{m}, w)$.

Definition C.E.1 (SEM-SO-CCA2) A public key cryptosystem (G, E, D) is SEM-SO-CCA2 secure if, for any PPT message distribution $\mathcal{M}$, any PPT relations $\mathcal{R}$ any PPT adversary $\mathcal{A}$, there is a simulator $S = (S_1, S_2)$ s.t. the outcome of real and ideal games are identical with all but negligible probability, *i.e.*,

$$\Pr[\mathsf{sem\text{-}cca2\text{-}real} \neq \mathsf{sem\text{-}cca2\text{-}ideal}] \leq \nu.$$

For some negligible function ν.

The notion of SEM-SO-CCA1 security is defined by means of similar experiments, but no decryption query is allowed after the selective opening query in the real game.

Similarly to the indistinguishability case, we remark that, if the adversary is not allowed to make decryption queries at all, this notion reduces to SEM-SO-ENC security.

C.E.1 Unduplicatable Set Selection

Unduplicatable set selection was used implicitly in [NY90] and [DIO98], and formalized in [Sah99]. The description below is essentially that of [Sah99].

The goal of unduplicatable set selection is to create a mapping from $\mathfrak{g} : \{0,1\}^k \to B$ such that, for all distinct $a^1, \ldots, a^n, a^{n+1} \in \{0,1\}^k$,

$$\mathfrak{g}(a^{n+1}) \not\subset \bigcup_{i=1}^{n} \mathfrak{g}(a^i).$$

In [Sah99], Sahai gives a simple construction based on polynomials which we recall here. Let $\ell = 2^{\lceil \log_2 2nk \rceil}$, so $\ell > 2nk$, and let $Y = \mathbb{F}_\ell \times \mathbb{F}_\ell$, and $B \subset \mathcal{P}(Y)$. To each $a \in \{0,1\}^k$, we may associate a polynomial

$$f_a(x) = a_0 + a_1 x + \cdots a_{k-1} x^{k-1} \in \mathbb{F}_\ell[x].$$

Then, if we set

$$\mathfrak{g}(a) = \{(t, f_a(t)) : t \in \mathbb{F}_\ell\} \subset Y,$$

we have $|\mathfrak{g}(a)| = \ell$ and, if $a \neq a'$, it holds that $|\mathfrak{g}(a) \cap \mathfrak{g}(a')| \leq k - 1$. Thus,

$$\left|\mathfrak{g}(a^{n+1}) \setminus \bigcup_{i=1}^{n} \mathfrak{g}(a^i)\right| = \left|\mathfrak{g}(a^{n+1}) \setminus \bigcup_{i=1}^{n} \mathfrak{g}(a^{n+1}) \cap \mathfrak{g}(a^i)\right|$$
$$\geq \left|\mathfrak{g}(a^{n+1})\right| - \sum_{i=1}^{n} \left|\mathfrak{g}(a^{n+1}) \cap \mathfrak{g}(a^i)\right| \geq \ell - n(k-1) \geq \frac{\ell}{2}.$$

We call $\mathfrak{g}$ an (n, k)-unduplicatable set selector.

C.E.2 Non-Interactive Zero-Knowledge

One of the most successful techniques for securing cryptosystems against chosen-ciphertext attacks has been the Naor-Yung paradigm [NY90]. Roughly said, the idea is to encrypt the message twice and include a non-interactive zero-knowledge (NIZK) proof that both encryptions encrypt the same plaintext. The proof of security then uses the NIZK simulator to simulate the proof for the challenge ciphertext. This method has since been refined in [DDN91,Sah99,DDO+01,Lin06] (among others).

Our construction of SEM-SO-CCA1 encryption follows the general Naor-Yung paradigm [NY90]. However, the selective opening of the encryption query poses new challenges. In particular, if we naively try to apply the Naor-Yung technique, we immediately encounter difficulties because our challenger must reveal the messages and randomness for half of the ciphertexts in the challenge. This will immediately reveal to the adversary that the proofs were simulated. It requires new ideas to overcome this difficulty.

We now give a brief definition of the properties of a non-interactive zero-knowledge proof of knowledge with honest-prover state reconstruction (originally defined and constructed in [GOS06a]).

Let $\mathcal{R}$ be an efficiently computable binary relation and let $L = \{x : \exists w \text{ such that } (x, w) \in \mathcal{R}\}$. We refer to L as a language, x as a statement, and w as a witness. A non-interactive proof system for L is a triple of PPT algorithms $(\mathsf{CRSgen}, \mathsf{Prover}, \mathsf{Verifier})$ such that

- $\sigma \leftarrow \mathsf{CRSgen}(1^\lambda)$: generates a common reference string σ.
- $\pi \leftarrow \mathsf{Prover}(\sigma, x, w)$: given x and a witness w for x s.t. $\mathcal{R}(x, w) = 1$, the Prover outputs a proof π.
- $b \leftarrow \mathsf{Verifier}(\sigma, x, \pi)$: on inputs x and a purported proof π, $\mathsf{Verifier}$ outputs a bit $b \in \{0, 1\}$.

Definition C.E.2 A triple $(\mathsf{CRSgen}, \mathsf{Prover}, \mathsf{Verifier})$ is called a non-interactive zero-knowledge (NIZK) proof of knowledge with honest-prover state reconstruction if it satisfies the following properties

- **Completeness:** For all adversaries $\mathcal{A}$, there exists a negligible function ν such that

$$\Pr\left[\begin{array}{l}\sigma \leftarrow \mathsf{CRSgen}(1^\lambda); (x, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow \mathsf{Prover}(\sigma, x, w) \\ \mathsf{Verifier}(\sigma, x, \pi) = 1 \text{ if } (x, w) \in \mathcal{R}\end{array}\right] > 1 - \nu.$$

- **Soundness:** For all adversaries $\mathcal{A}$, there is a negligible function ν such that

$$\Pr\left[\sigma \leftarrow \mathsf{CRSgen}(1^\lambda); (x, \pi) \leftarrow \mathcal{A}(\sigma) : \mathsf{Verifier}(\sigma, x, \pi) = 0 \text{ if } x \notin L\right] > 1 - \nu.$$

- **Knowledge Extraction:** There is an extractor $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ such that, for all adversaries $\mathcal{A}$,

$$\left|\Pr\left[\sigma \leftarrow \mathsf{CRSgen}(1^\lambda) : \mathcal{A}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau) \leftarrow \mathsf{Ext}_1(1^\lambda) : \mathcal{A}(\sigma) = 1\right]\right| < \nu,$$

and

$$\Pr\left[\begin{array}{l}(\sigma,\tau) \leftarrow \mathsf{Ext}_1(1^\lambda); (x,\pi) \leftarrow \mathcal{A}(\sigma); w \leftarrow \mathsf{Ext}_2(\sigma,\tau,x,\pi) \\ \mathsf{Verifier}(\sigma,x,\pi) = 0 \text{ or } (x,w) \in \mathcal{R}\end{array}\right] > 1 - \nu$$

For some negligible function ν.

- **Zero-Knowledge:** There exists a simulator $S = (S_1, S_2)$, such that for all adversaries $\mathcal{A}$,

$$\left|\Pr\left[\sigma \leftarrow \mathsf{CRSgen}(1^\lambda) : \mathcal{A}^{\mathsf{Prover}(\sigma,\cdot,\cdot)}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau) \leftarrow S_1(1^\lambda) : \mathcal{A}^{S'(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right]\right| < \nu,$$

where S' is defined

$$S' = \begin{cases} S_2(\sigma,\tau,x) & \text{if } (x,w) \in \mathcal{R}, \\ \perp & \text{otherwise.}\end{cases}$$

- **Honest-Prover State Reconstruction:** There exists a simulator $\mathsf{SR} = (\mathsf{SR}_1, \mathsf{SR}_2, \mathsf{SR}_3)$ such that for all adversaries $\mathcal{A}$

$$\left|\Pr\left[\sigma \leftarrow \mathsf{CRSgen}(1^\lambda); \mathcal{A}^{\mathsf{Prover}(\sigma,\cdot,\cdot)}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau) \leftarrow \mathsf{SR}_1(1^\lambda); \mathcal{A}^{\mathsf{SR}(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right]\right| < \nu,$$

where $\mathsf{Prover}(\sigma,x,w)$ samples $r \leftarrow \mathsf{coins}(\mathsf{Prover})$, sets $\pi = \mathsf{Prover}(\sigma,x,w,r)$ and returns (π, r) whereas SR samples $r^* \leftarrow \mathsf{coins}(\mathsf{SR}_2)$, sets $\pi' = \mathsf{SR}_2(\sigma,\tau,x,r^*)$ and finally SR sets $r' \leftarrow \mathsf{SR}_3(\sigma,\tau,x,w,r^*)$ and returns (π', r'). Both oracles output $\perp$ if $(x,w) \notin \mathcal{R}$.

C.E.3 A SEM-SO-CCA1 Construction Based on the Naor-Yung Paradigm

Along with NIZK proofs with honest-prover state reconstruction, our construction relies on a number of common cryptographic tools. We will also require a strongly unforgeable one-time signature scheme. In the SEM-SO-CCA1 game, a single encryption query is actually n separate encryptions and we will require an unduplicatable set selector $\mathfrak{g}$ for sets of size n (see Appendix C.E.1 for a description of unduplicatable set selectors). Finally, we will require a lossy encryption scheme with efficient opening.

While the construction outlined below uses a one-time signature scheme (as in [DDN91]), the signature scheme can be removed and replaced by a strictly combinatorial construction as in [NY90]. We note that, although our construction is similar to the IND-CCA2 construction of [DDN91], the proof of SEM-SO-CCA1 security *does not* extend to SEM-SO-CCA2 security because the adversary learns the signing keys used for half of the ciphertexts in the challenge query, which allows her to create arbitrary signatures corresponding to those verification keys. This appears to be a significant problem when trying to adapt many of the known IND-CCA2 constructions to the IND-SO-CCA2 or SEM-SO-CCA2 settings.

Let $\Pi^{so} = (G_{so}, E, D)$ be an efficiently openable (and thus SEM-SO-ENC secure) lossy cryptosystem. Let $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver})$ be a strongly unforgeable one-time signature scheme where the public key space in contained in $\{0,1\}^\lambda$. Let $\mathfrak{g}$ be an (n,λ)-unduplicatable set selector and let $\ell = |\mathfrak{g}(0^\lambda)|$ and $L = \mathfrak{g}(\{0,1\}^\lambda)$.

Let $(\mathsf{CRSgen}, \mathsf{Prover}, \mathsf{Verifier})$ be a NIZK proof of knowledge with honest-prover state reconstruction for the language given by the relation $((e_0,e_1),(m,r_0,r_1)) \in \mathcal{R}$ if $e_0 = E(m,r_0)$ and $e_1 = E(m,r_1)$.

Our SEM-SO-CCA1 scheme works as follows.

- **KeyGen:** Generate two key pairs for Π^{so} and reference strings for the NIZK proof system

$$(pk_0, sk_0) \leftarrow G_{so}(1^\lambda),\ (pk_1, sk_1) \leftarrow G_{so}(1^\lambda),\ \text{and } \sigma_i \leftarrow \mathsf{CRSgen}(1^\lambda) \text{ for } i \in L.$$

  Set $pk = (pk_0, pk_1, \{\sigma_i\}_{i\in L})$ and $sk = (sk_0, sk_1)$.

- **Encryption:** Pick random coins

$$r^{sig} \leftarrow \mathsf{coins}(\mathsf{Sign}),\ r_0 \leftarrow \mathsf{coins}(E),\ r_1 \leftarrow \mathsf{coins}(E),\ r_i^{nizk} \leftarrow \mathsf{coins}(\mathsf{Prover}) \text{ for } i = 1, \ldots, \ell.$$

  Generate keys $(vk, sk) = \mathsf{G}(r^{sig})$ for a one-time signature using randomness r^{sig}.

  To encrypt a message m, calculate

$$e_0 = E(pk_0, m, r_0), \quad e_1 = E(pk_1, m, r_1).$$

  Using the witness $w = (m, r_0, r_1)$, generate NIZK proofs

$$\overline{\pi} = (\pi_1, \ldots, \pi_\ell) = (\mathsf{Prover}(\sigma_i, (e_0, e_1), w))_{i\in\mathfrak{g}(vk)}$$

  using r_i^{nizk} in the i^{th} iteration of Prover. Generate a signature $\mathsf{sig} = \mathsf{Sign}(e_0, e_1, \overline{\pi})$ and output

$$c = (vk, e_0, e_1, \overline{\pi}, \mathsf{sig}).$$

- **Decryption:** Given a ciphertext $c = (vk, e_0, e_1, \overline{\pi}, \mathsf{sig})$, check that $\mathsf{Ver}(vk, (e_0, e_1, \overline{\pi})) = 1$, and return $\perp$ otherwise. For each $i \in \mathfrak{g}(vk)$, check that $\mathsf{Verifier}(\sigma_i, (e_0, e_1), \pi_i) = 1$ and return $\perp$ otherwise. If all checks are successful, return $m = D(sk_0, e_0)$.

Theorem C.E.3 This scheme is SEM-SO-CCA1 secure.

Proof: We will show how to use an adversary $\mathcal{A}$ in the sem-cca1-real game to construct a simulator for the sem-cca1-ideal game. To do this, we begin by considering a series of games.

- Game_0: is the actual sem-cca1-real game.

- Game_1: is as Game_0 but the verification keys $(vk^{chal,1}, sk^{chal,1}), \ldots, (vk^{chal,n}, sk^{chal,n})$ to be used in the challenge ciphertexts are chosen during the parameter generation phase. In addition, we raise a failure event F_1, which is the occurrence of a decryption query $(vk, e_0, e_1, \overline{\pi}, \mathsf{sig})$ such that $vk = vk^{chal,j}$ for some $j \in \{1, \ldots, n\}$.

- Game_2: is identical to Game_1 but the common reference strings are now generated as

$$\sigma_i = \begin{cases} \sigma \leftarrow \mathsf{CRSgen}(1^\lambda) & \text{if } i \in \mathfrak{g}(vk^{chal,j}) \text{ for some } j \in [n] \\ \text{the first output of } (\sigma, \tau) \leftarrow \mathsf{Ext}_1(1^\lambda) & \text{otherwise.} \end{cases}$$

  In addition, to handle decryption queries $(vk, e_0, e_1, \overline{\pi}, \mathsf{sig})$, we now use any index $i \notin \mathfrak{g}(vk) \in \{1, \ldots, \ell\}$ to recover (m, r_0, r_1) from the proof π_i using the trapdoor τ_i of the extractable reference string σ_i. Such an index $i \in \{1, \ldots, \ell\}$ must exist since $\mathfrak{g}(vk) \not\subset \bigcup_{j=1}^n \mathfrak{g}(vk^{chal,j})$.

- Game_3 in this game, we switch both pk_0 and pk_1 to the lossy mode and proceed as in Game_2.

- Game_4: we now use the honest-prover state reconstruction simulator $\mathsf{SR} = (\mathsf{SR}_1, \mathsf{SR}_2, \mathsf{SR}_3)$. We first bring a new change to the generation of reference strings at the beginning of the game. Namely, for each $i \in L$ such that $i \in \mathfrak{g}(vk^{chal,j})$, for some $j \in [n]$, we set $(\sigma_i, \tau_i) \leftarrow \mathsf{SR}_1(1^\lambda)$. Also, in the generation of target ciphertexts, we ignore the witnesses and simulate the "proofs"

$$\overline{\pi} = \{\pi_i\}_{i \in \mathfrak{g}(vk^{chal,j})} = \{\mathsf{SR}_2(\sigma_i, \tau_i, (e_0, e_1), r_i^*)\}_{i \in \mathfrak{g}(vk^{chal,j})},$$

  for each $i \in \{1, \ldots, \ell\}$, $j \in \{1, \ldots, n\}$. Also, when the adversary asks for the opening of a subset of the target ciphertexts, we use the honest-prover state reconstructor to generate

$$r_i \leftarrow \mathsf{SR}_3(\sigma_i, \tau_i, (e_0, e_1), (m, r_0, r_1, r_i^*)),$$

  and return these r_i (instead of the coins r_i^* that were actually used to simulate proofs).

- Game_5: in this game, the challenger generates all target ciphertexts as encryptions of a dummy message ξ. In addition, the choice of $\mathbf{m} \xleftarrow{\$} \mathcal{M}$ is postponed until the moment of the opening query. When $\mathcal{A}$ asks for the opening of a subset of the target ciphertexts, we use the efficient openability of (G_{so}, E, D) to generate $\{r_i\}_{i \in I}$ that explain $\mathbf{m}[I]$. Otherwise, the simulator proceeds as in Game_4.

Let W_i be the distribution of the adversary's output in game i. Clearly, W_0 is almost identical to W_1 since, given that $vk^{chal,1}, \ldots, vk^{chal,n}$ are independent of the adversary's view until the challenge phase, the failure event F_1 occurs with probability smaller than $qn\delta$ if q is the number of decryption queries and δ is the maximal probability for a given verification key to be generated by G. In other words, we only need the property that vk is unpredictable and we could use a simple combinatoric argument as in [NY90]. However, a one-time signature scheme clearly has this property as well.
To show that W_1 and W_2 are only negligibly different, notice that, by the unduplicatability of $\mathfrak{g}$, there will always be at least one valid proof generated with an extractable CRS. Hence, we will always be able to answer decryption queries. It comes that any significant difference between Game_2 and Game_1 would imply the ability of the adversary to break either the soundness or the knowledge extraction property of the proof system. By virtue of the latter's security, W_2 must be negligibly close to W_1.
Since the challenger never uses the decryption keys corresponding to pk_0 and pk_1 in Game_2 (instead the challenger decrypts with the knowledge extractor), the distributions W_2 and W_3 must be computationally indistinguishable. Otherwise, the challenger could distinguish injective keys from lossy keys in the underlying lossy encryption scheme (G_{so}, E, D).
Now, it is easy to see that any PPT adversary that can distinguish between Game_3 and Game_4 can be used to distinguish honestly generated proofs for the real CRS of Game_3 and the outputs of the honest-prover reconstruction simulator $(\mathsf{SR}_1, \mathsf{SR}_2, \mathsf{SR}_3)$ (really $n\ell$ such simulators) in Game_4. Such an adversary indeed breaks the indistinguishability of the honest-prover state reconstruction simulator, losing a factor of $n\ell$ (because we are making $n\ell$ comparisons).
Finally, we also note that, for each challenge ciphertext, SR_2 generates proofs without using witnesses and, since pk_0 and pk_1 are both lossy keys, each challenge ciphertext is statistically independent of the plaintext. Moreover, since Π^{so} allows for efficient opening under lossy keys, the challenger can open any such ciphertext to any desired plaintext without affecting $\mathcal{A}$'s view. It comes that the statistical distance between W_5 and W_4 is negligible.
Thus, we have shown that, for any efficient adversary $\mathcal{A}$, the value of Game_0 will be computationally indistinguishable from the value of Game_5. Now, we show how to use the adversary of Game_5 to build a simulator for the sem-cca1-ideal game.

Specifically, the simulator runs $\mathcal{A}$ internally exactly as Game_5 does. In particular, it generates lossy keys pk_0, pk_1 and reference strings on its own and answers decryption queries as in Game_2-Game_5. When $\mathcal{A}$ asks for a subset I, the simulator asks for openings of the same subset I. Using $\{m_i\}_{i\in I}$, the simulator runs the efficient opening procedure of (G_{so}, E, D) to generate $\{r_i\}_{i\in I}$. As in Game_5, the simulator then uses the state reconstructor SR_3 to generate randomness that look like an honest prover's random coins for the witnesses $\{(m_i, r_i)\}_{i\in I}$. Finally, when $\mathcal{A}$ outputs w, the simulator outputs the same w. Since $\mathcal{A}$'s output in Game_5 is indistinguishable from her output in the sem-cca1-real game, the output of the simulator will be indistinguishable from $\mathcal{A}$'s output in the sem-cca1-real game.

∎

A similar argument shows that this construction will be IND-SO-CCA1 if the underlying encryption scheme is IND-SO-ENC instead of SEM-SO-ENC secure.

Notice, however, that if we consider the SEM-SO-CCA2 game, then Game_1 and Game_2 are distinguishable. This is because when an adversary gets an opening of one of the challenge ciphertexts, he also receives the secret key of the one-time signature used on that message. He can thus sign any message using that verification key. This is the primary stumbling block when trying to build SEM-SO-CCA2 (or IND-SO-CCA2) encryptions using one-time signature schemes.

C.F The Paillier Cryptosystem

We briefly review the Paillier cryptosystem [Pai99] that was extended by Damgård and Jurik [DJ01]. The cryptosystem works over $\mathbb{Z}^*_{N^2}$. From the Binomial Theorem, we have

$$(1+N)^a = 1 + aN \mod N^2,$$

so $(1+N)$ generates a cyclic subgroup of order N. In this group, we can compute "partial" discrete logarithms efficiently by $L(x) = \frac{x-1}{N}$, since $L((1+N)^a) = L(1+aN) = a$. Now, if g generates $\langle 1+N\rangle$ and $c = g^a \bmod N^2$, we have $a = L(c)L(g)^{-1} \bmod N$.

- **Parameter Generation:**
  - Generate primes p, q of length $\lambda/2$ and sets $N = pq$.
  - Generate $g \in \mathbb{Z}^*_{N^2}$ such that N divides the order of g.
    This condition is easy to verify if you have the factorization of N.

  The public parameters are $pk = (N, g)$. The secret key is $sk = \text{lcm}(p-1, q-1)$.

- **Encryption:** to encrypt $m \in \mathbb{Z}_N$, chooose $r \xleftarrow{\$} \mathbb{Z}^*_N$ (r is actually drawn in $\mathbb{Z}_N$, but the distributions are statistically close) and compute $c = E(pk, m, r) = g^m r^N \mod N^2$.

- **Decryption:** given a ciphertext $c \in \mathbb{Z}^*_{N^2}$,

$$m = \frac{L(c^{sk} \bmod N^2)}{L(g^{sk} \bmod N^2)} \mod N.$$

This cryptosystem is IND-CPA secure under the Decisional Composite Residuosity assumption (DCR), which (informally) says the following.

Assumption C.F.1 Decisional Composite Residuosity (DCR): If $N = pq$ is an λ-bit RSA modulus,

$$\{g \leftarrow \mathbb{Z}^*_{N^2} : \ g\} \approx_c \{g \leftarrow \mathbb{Z}^*_{N^2} : \ g^N\}.$$

Groth-Sahai Proof System and Applications

Appendix D:

Short Blind Signatures

Journal of Computer Security 2013

Olivier Blazy, Georg Fuchsbauer, David Pointcheval and Damien Vergnaud

This paper presents (partially) blind signature schemes, in which the number of interactions between the user and the signer is minimal and whose blind signatures are short. Our schemes are defined over bilinear groups and are proved secure in the common-reference-string model without random oracles and under standard assumptions. It also extends Waters signatures to non-binary alphabets by proving a new result on the underlying hash function.

Appendix E:

Fair Blind Signatures without Random Oracles

Africacrypt 2010

Georg Fuchsbauer and Damien Vergnaud

*This paper revisit the notion of fair blind signatures (*i.e. *blind signatures with revocable anonymity and unlinkability). It provides a new security model for fair blind signatures (reinforcing the model given by Hufschmitt and Traoré in 2007) and gives the first practical fair blind signature scheme with a security proof in the standard model.*

Appendix F:

Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model

CANS 2009

Benoît Libert and Damien Vergnaud

This paper presents the first efficient verifier-local revocation group signature – where revocation messages are only sent to signature verifiers – providing backward unlinkability *(*i.e. *previously issued signatures remain anonymous even after the signer's revocation) with a security proof in the standard model.*

Appendix D

Short Blind Signatures

Journal of Computer Security 2013
[BFPV13] with O. Blazy, G. Fuchsbauer and D. Pointcheval

Abstract : *Blind signatures allow users to obtain signatures on messages hidden from the signer; moreover, the signer cannot link the resulting message/signature pair to the signing session. This paper presents blind signature schemes, in which the number of interactions between the user and the signer is minimal and whose blind signatures are short. Our schemes are defined over bilinear groups and are proved secure in the common-reference-string model without random oracles and under standard assumptions: CDH and the decision-linear assumption. (We also give variants over asymmetric groups based on similar assumptions.) The blind signatures are Waters signatures, which consist of 2 group elements.*
Moreover, we instantiate partially blind signatures, where the message consists of a part hidden from the signer and a commonly known public part, and schemes achieving perfect blindness. We propose new variants of blind signatures, such as signer-friendly *partially blind signatures, where the public part can be chosen by the signer without prior agreement, 3-party blind signatures, as well as blind signatures on multiple aggregated messages provided by independent sources.*
We also extend Waters signatures to non-binary alphabets by proving a new result on the underlying hash function.

D.1 Introduction

Blind signature schemes were proposed by Chaum in 1982 [Cha82]. They define an interactive signature protocol between a user and a signer, guaranteeing that the signed message, and even the resulting signature, are unknown to the signer; this property is called *blindness*. More precisely, if the signer runs several executions of the protocol leading to several message/signature pairs, he cannot link a pair to a specific execution: the view of the signer is unlinkable to the resulting message/signature pair. This unlinkability can be either computational, in which case we talk about *computational blindness*, or information-theoretic, we then talk about *perfect blindness*. The second security property for blind signatures is a notion of unforgeability, which has been formalized by Pointcheval and Stern [PS00] motivated by the use of blind signatures for e-cash: a user should not be able to produce more message/signature pairs (coins) than the number of signing executions with the bank (withdrawals). More recently, Schröder and Unruh [SU12] revisited the security model for other contexts.

There have been several constructions with highly interactive blind-signing protocols (like [Oka06]), before Fischlin [Fis06] gave a generic construction of *round-optimal* blind signature schemes, where there is only one round of communication between the user and the signer. Abe et al. [AFG+10] have efficiently instantiated his blueprint, in which the user obtains a signature on the blinded message from the signer, but in order to achieve unlinkability, a blind signature is defined as a non-interactive zero-knowledge proof of knowledge of this signature, leading to an increase of the signature size.

In [BFPV11, BPV12b], we presented a new approach, where instead of proving knowledge of it, the user merely *randomizes* the signature, which suffices to achieve unlinkability. Blind signatures are thus signatures of the underlying scheme, which are much shorter than proofs of knowledge thereof. In our construction, the underlying (and thus the blind) signatures are Waters signatures [Wat05], which consist of 2 elements from a bilinear group, while the message is a scalar. In comparison, the most efficient scheme by Abe et al. [AFG+10] has messages consisting of two group elements, while a signature consists of 18+16 (in $\mathbb{G}_1$ and $\mathbb{G}_2$) group elements. Furthermore, our schemes are secure under the, meanwhile standard, "decision linear" (DLin) [BBS04] assumption, while the schemes in [AFG+10] are based on newly introduced "q-type" assumptions. The drawback of our scheme is that, while being round-optimal, the user must send much more information to the signer during the blind signature-issuing protocol. While all round-optimal schemes mentioned so far are proven secure in the Common Reference String (CRS) model, round-optimal blind signatures without CRS have been proposed by Garg et al. [GRS+11], who however only give impractical generic constructions.

A loophole in standard blind signatures was first identified by Abe and Okamoto [AO00]: the signer has no control at all over which messages are signed. In classical e-cash schemes, unforgeability, which restricts a user's number of coins to the number of withdrawals, was sufficient. For the case that the bank wants to include an expiration date in the message (in addition to the user-chosen serial number), Abe and Fujisaki [AF96] propose *partially blind signatures*, where the user and the signer agree on part of the message before running the blind signing protocol.

The above-mentioned scheme from [AFG+10] was extended to partially blind signature scheme in [Fuc11]. More recently, Seo and Cheon [SC12] presented a construction leading to (partially) blind-signature schemes in the CRS model. However, their construction relies on a trick consisting in starting from prime-order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ and considering group elements in $\mathcal{G} = \mathbb{G}_1 \oplus \mathbb{G}_2 \oplus \mathbb{G}_3$. While their approach provides nice theoretical tools, the resulting signatures lie in $\mathcal{G}^2$ and are therefore three times longer than our proposal.

Our contributions. In this paper, we extend our earlier results from [BFPV11] in several directions. Instead of using an encryption scheme to blind the message to be signed, we use a *mixed commitment scheme* [DN02] in which commitments can be set up to either be perfectly binding (like encryption) or perfectly hiding.

We first present a blind signature scheme with perfect blindness, using the perfectly hiding setup of Groth-Sahai commitments [GS08]. We then extend the model of partially blind signatures to avoid prior agreement on the public part of the message between signer and user: signers can decide on its content only before sending their message in the signature-issuing protocol. (If the user does not agree with the public part, they can simply discard the signature and start anew.) We call this new primitive *signer-friendly partially blind signatures*. While of course not forbidding any prior agreement on the public part, this primitive offers an effectively round-optimal protocol (as there need not be any communication between the user and signer before running the protocol itself).

Using perfectly hiding commitments, we present a round-optimal signer-friendly partially

blind signature with perfect blindness. Again, instead of having a computational overhead for the agreement, both signer and user can simply choose their contribution on the fly. The signer can always refuse to sign when the user's public information does not suit him, and the user can always choose to discard irrelevant signatures.

Using the perfectly binding setting for the mixed commitment (and thus discarding perfect blindness) we take advantage of the fact that user and signer can independently choose their inputs and consider a new context: the message to be signed is an aggregation of inputs that come from several independent sources which cannot communicate with each other. We consider several aggregation procedures. First we present a way to obtain a signature on the concatenation of the inputs and then present a shorter instantiation yielding a signature on the sum of the inputs messages. Addition of messages in often used, *e.g.* when counting votes, or aggregating sensor information, *etc.*

All our constructions are based on the Waters signature [Wat05], which signs messages that are binary strings. We reconsider the programmable hash function used for these signatures over a *non-binary* alphabet, in a similar way to what Hofheinz and Kiltz [HK08] did for the binary case. We prove a negative result on the $(2, 1)$-programmability, but a positive one on the $(1, poly)$-programmability of this hash function. The latter result immediately yields Waters signatures over a non-binary alphabet, which in turn leads to a reduction in the number of public-key elements.

Instantiations. We give several instantiations of our different blind-signature schemes, all of which are based on weak assumptions. Our constructions mainly use the following two building blocks, from which they inherit their security: Groth-Sahai proofs for languages over pairing-friendly groups [GS08] and Waters signatures derived from the scheme in [Wat05] and first used in [BW06b]. Since verification of the revisited Waters signatures from [BFPV11] is a statement of the language for Groth-Sahai proofs, these two building blocks combine smoothly. Moreover, Waters signatures (and its variant) are fully randomizable thanks to the homomorphic property of its random coins. The first instantiations are over pairing-friendly elliptic curves and use linear commitments. Both unforgeability and blindness of these constructions rely solely on the decision linear assumption. An instantiation with improved efficiency, in *asymmetric* bilinear groups, using the SXDH variant of Groth-Sahai proofs and commitments, is drafted in the Appendix D.A. This setting requires Waters signatures over asymmetric groups, which in [BFPV11] we proved secure under a slightly stronger assumption, termed CDH^+, which in symmetric groups is identical to CDH.

Applications. The properties of our blind signature schemes find various kinds of applications for anonymity:

E-voting. The security of several e-voting protocols (see [BT94, CMFP$^+$10, DK10]) relies on the fact that each ballot is certified by an election authority. Since this authority should not learn the votes, a blind signature scheme (or a partially blind one, if for example the authority wants to specify the election in the ballot) is usually used to achieve this property. Using a perfectly blind scheme, privacy is even achieved in an information-theoretic sense. Our scheme is the first to achieve this property without random oracles and under standard complexity assumptions.

E-cash. As mentioned above, partially blind signatures play an important role in many electronic-commerce applications. In e-cash systems, for instance, the bank issuing coins must ensure that the message contains accurate information such as the face value of the e-coin without seeing it. Moreover, in order to prevent double-spending, the bank's database has to record all spent coins. Partially blind signatures can cope with these requirements, since the bank can explicitly include some information such as the face value or the expiration date in the coin. The latter, for example, can be included in the coin by the bank without prior agreement with the client.

Data aggregation in networks. A wireless (ad hoc) sensor network (WSN) consists of many sensor nodes that are deployed for sensing the environment and collecting data from it. Since transmitting and receiving data are the most energy-consuming operations, data aggregation has been put forward as an essential paradigm in these networks. The idea is to combine the data coming from different sources, minimizing thus the number of transmissions and saving energy. In this setting a WSN usually consists of three types of nodes:

- *sensor nodes*, which are small devices equipped with one or more sensors, a processor and a radio transceiver for wireless communication;
- *aggregation nodes* (or aggregators) performing the data aggregation (*e.g.* average, sum, minimum or maximum of data); and
- *base stations*, responsible for querying the nodes and gathering the data collected by them.

WSNs are at high security risk and two important security goals when doing in-network data aggregation are *data confidentiality* and *data integrity*. When homomorphic encryption is used for data aggregation, end-to-end encryption allows aggregation of the encrypted data and thus end-to-end data confidentiality, as aggregators never decrypt any data. Achieving data integrity is a harder problem and the attack where a sensor node reports a false reading value is typically not considered (the impact of such an attack being usually limited). The main security threat is a *data-pollution attack* in which an attacker tampers with an aggregation node to make the base station receive wrong aggregated results.

While in most conventional data-aggregation protocols, data integrity and privacy are not preserved at the same time, our multi-source blind signature primitive yields data confidentiality and prevents data-pollution attacks simultaneously by using the following simple protocol:

1. Data aggregation is initiated by a base station, which broadcasts a query to the whole network.
2. Sensor nodes then report values of their readings which are encrypted under base station's public key to their aggregators.
3. The aggregators perform data aggregation via the homomorphic properties of the encryption scheme, (blindly) sign the result and route the aggregated results back to the base station.
4. The base station decrypts the aggregated data and the signature, which proves the validity of the gathered information to the base station (but also to any other third party).

D.2 Definitions

This section presents successively the various tools we need to describe the global framework and the security model for partially blind signature schemes. There are two different layers for the construction, one relying on commitments and the other on signatures.

D.2.1 Commitments

The original construction from [BFPV11] uses an encryption scheme $\mathcal{E}$ described via four polynomial-time algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$):

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme;

- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key ek and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; \rho)$ outputs an encryption c of the message m under the encryption key ek with randomness $\rho \in \mathcal{R}_e$;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ outputs the plaintext m, encrypted in the ciphertext c or $\perp$ in case of an invalid ciphertext.

Such an encryption scheme is required to have the following security properties:

- *Correctness*: For every key pair $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every message m, and every random ρ, we should have

$$\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m; \rho)) = m \ .$$

- *Indistinguishability under Chosen-Plaintext Attack [GM84]*: This notion, formalized by the adjacent game, states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two candidate messages.

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND}, \mathsf{ek})$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS}, c^*)$
6. `RETURN` b'

We define the adversary's advantage as:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] \ .$$

To generalize the original construction from [BFPV11], we will replace encryption by a mixed commitment scheme [DN02]. A commitment scheme allows anyone to *commit* to a message m by running Commit on a key ck, m and some randomness ρ. The resulting commitment c should not reveal anything about the committed message, *i.e.*, the commitment is *hiding*. By revealing the randomness ρ, the commitment can be *opened*, as anyone can run $c' \leftarrow \mathsf{Commit}(\mathsf{ck}, m; \rho)$ and check whether $c' = c$. However, the commitment should be *binding*, in that for every c there is only a single m to which c can be opened (in that for all $m \neq m', \rho, \rho'$: $\mathsf{Commit}(\mathsf{ck}, m; \rho) \neq \mathsf{Commit}(\mathsf{ck}, m'; \rho')$).

In a *mixed commitment* scheme, the *commitment keys* ck can be set up in two ways, which are computationally indistinguishable. One type leads to *perfectly hiding* commitments, that is, the commitment does not contain any information about the committed value; whereas the second type leads to *perfectly binding* commitments, that is, not even an unbounded adversary can find a commitment and two different openings for it. We moreover require that there be a trapdoor which enables efficient extraction of the committed value.

Of course, when applied, one will use only one kind of setup depending on the main security goal or the properties demanded of the final scheme.

Definition D.2.1 [Mixed Commitment Scheme [DN02]] A *mixed commitment scheme* $\mathcal{C}$ is a 5-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{WISetup}, \mathsf{ExSetup}, \mathsf{Commit}, \mathsf{Extract})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme, and more specifically the commitment key ck by running one of the following algorithms:

  - $\mathsf{WISetup}(1^\lambda)$, outputs a perfectly hiding commitment key ck;
  - $\mathsf{ExSetup}(1^\lambda)$, outputs a perfectly binding commitment key ck and an *extraction key* xk (to be kept secret);

- $\mathsf{Commit}(\mathsf{ck}, m; \rho)$, outputs a commitment c of a message m under a commitment key ck and a random $\rho \xleftarrow{\$} \mathbb{Z}$;

- $\mathsf{Extract}(\mathsf{xk}, c)$, if c is a commitment under a binding key ck, created together with xk, it outputs m.

We sometimes write $\mathsf{Decommit}(\mathsf{ck}, c, m; \rho)$, which verifies correct opening of a commitment by checking whether $c = \mathsf{Commit}(\mathsf{ck}, m; \rho)$.

We require that the two setups be indistinguishable, that is, given a commitment key ck, it should be hard to decide whether the key has been generated by $\mathsf{WISetup}$ or $\mathsf{ExSetup}$: for any polynomial-time adversary $\mathcal{A}$ receiving a key ck from either distribution, its advantage in distinguishing which distribution it was should be negligible in λ.

Note that by indistinguishability of the setups, perfectly hiding commitments are automatically computationally binding and perfectly binding commitments are automatically computationally hiding.

D.2.2 Signatures

We now review several signature primitives, from classical to blind signatures.

Definition D.2.2 [Signature Scheme] A *signature scheme* $\mathcal{S}ig$ is a 4-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme;

- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public (verification) key vk and the private (signing) key sk;

- $\mathsf{Sign}(\mathsf{sk}, m; s)$ on input a signing key sk, a message m from the message space $\mathcal{M}$ and random coins $s \in \mathcal{R}_s$, produces a signature σ.

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

We expect signatures to be *existentially unforgeable under chosen-message attacks* [GMR88]: even after querying n valid signatures on chosen messages $\{m_i\}_{i=1}^n$, an adversary should not be able to output a valid signature on a new message.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk} : \mathtt{OSign}(\mathsf{sk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5. `IF` $m^* \in \mathcal{SM}$ `THEN RETURN` 0
6. `ELSE RETURN` b

In $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}$, the adversary has access to the following oracle:

- $\mathtt{OSign}(\mathsf{sk}, m)$ chooses $s \xleftarrow{\$} \mathcal{R}_s$ and returns $\mathsf{Sign}(\mathsf{sk}, m; s)$, after adding m to the set of signed messages $\mathcal{SM}$.

The probability of success in this game is denoted by

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda) = 1] \ .$$

Definition D.2.3 [Randomizable Signature Scheme] Let $\mathcal{S}ig = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme with the following additional algorithm:

- $\mathsf{Random}(\mathsf{vk}, m, \sigma; s')$, on input a valid signature σ on a message m under vk, produces a new signature σ', again valid under vk on m, using the additional random coins $s' \in \mathcal{R}_s$.

A signature scheme is called *randomizable* if for any $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$, any message $m \in \mathcal{M}$, any random coins $s \in \mathcal{R}_s$, and the associated signature $\sigma = \mathsf{Sign}(\mathsf{sk}, m; s)$, the following two distributions are statistically indistinguishable:

$$\mathcal{D}_0 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Sign}(\mathsf{sk}, m; s')\} \qquad \mathcal{D}_1 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Random}(\mathsf{vk}, m, \sigma; s')\}$$

The usual unforgeability notions apply (except strong unforgeability, since the signature is malleable, by definition).

Definition D.2.4 [Blind Signature Scheme] A *blind signature scheme* $\mathcal{BS}$ is a 4-tuple of polynomial-time algorithms/protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param.
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- Signature Issuing is an interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, m)$, for a message $m \in \mathcal{M}$. It generates an output σ for the user: $\sigma \leftarrow \langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m) \rangle$.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ outputs 1 if the signature σ is valid w.r.t. m and vk, 0 otherwise.

The security of a blind signature scheme is defined through two different notions [PS00], unforgeability and blindness (see Figure D.1). An adversary $\mathcal{U}^*$ against the unforgeability tries to generate $q_s + 1$ valid signatures after at most q_s complete interactions with the honest signer. The blindness condition protects, on the other hand, against malicious signers. It states that a malicious signer $\mathcal{S}^*$ should be unable to decide which of two messages m_0, m_1 has been signed first in two executions (one for each message, hence the superscript "≤ 1" in $\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{BS},\mathcal{S}^*}$ in Figure D.1) with an honest user $\mathcal{U}$. Let σ_b be the signature on m_b. Note that the malicious signer $\mathcal{S}^*$ can choose the keys and thus the verification key vk given to users. However, if $\mathcal{S}^*$ refuses to sign one of the inputs (*i.e.* $\sigma_i = \perp$) then the two resulting signatures are set to $\perp$; the adversary therefore does not gain any advantage if he decides to prevent the normal game execution.

Our unforgeability notion slightly differs from the original one [PS00], in that we do not exclude malleability, as this could not be satisfied by the randomizable signatures we use. We thus count the number of distinct signed messages, which should not be larger than the number of interactions with the signer, whereas the original definition counted the number of distinct message/signature pairs: $\mathcal{BS}$ is *unforgeable* if, for any polynomial adversary $\mathcal{U}^*$ (malicious user), the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda)$ is negligible, where $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda) = 1]$, in the security game presented in Figure D.1. In this experiment, the adversary $\mathcal{U}^*$ can interact q_s times with the signing oracle $\mathcal{S}(\mathsf{sk}, \cdot)$ (hence the notation $\mathcal{U}^*(\mathsf{vk} : \langle \mathcal{S}(\mathsf{sk}, \cdot), \cdot \rangle^{\leq q_s})$) to execute the blind signature protocol: the adversary should not be able to produce more signatures on distinct messages than interactions with the signer. Our relaxation from the original *One-More Forgery* security is to accommodate randomizable signatures, for which from a message/ signature pair one can generate many signatures on the same message. This is the same difference as between classical and strong existential unforgeability for signatures.

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
$((m_1, \sigma_1), \ldots, (m_{q_s+1}, \sigma_{q_s+1})) \leftarrow \mathcal{U}^*(\mathsf{vk} : \langle \mathcal{S}(\mathsf{sk}, \cdot), \cdot \rangle^{\leq q_s})$;
$\mathtt{IF}\ \exists i \neq j : m_i = m_j\ \mathtt{OR}\ \exists i : \mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 0\ \mathtt{THEN\ RETURN}\ 0$
$\mathtt{ELSE\ RETURN}\ 1$

$\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{BS},\mathcal{S}^*}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk}, m_0, m_1, st_{\mathtt{FIND}}) \leftarrow \mathcal{S}^*(\mathtt{FIND}, 1^\lambda)$;
$b \leftarrow \{0, 1\}$;
$st_{\mathtt{ISSUE}} \leftarrow \mathcal{S}^*(\mathtt{ISSUE}, st_{\mathtt{FIND}} : \langle \cdot, \mathcal{U}(\mathsf{vk}, m_b) \rangle^{\leq 1}, \langle \cdot, \mathcal{U}(\mathsf{vk}, m_{1-b}) \rangle^{\leq 1})$;
$\mathtt{IF}\ \sigma_0 = \perp\ \mathtt{OR}\ \sigma_1 = \perp\ \mathtt{THEN}\ (\sigma_0, \sigma_1) \leftarrow (\perp, \perp)$;
$b^* \leftarrow \mathcal{S}^*(\mathtt{GUESS}, \sigma_0, \sigma_1, st_{\mathtt{ISSUE}})$;
$\mathtt{IF}\ b = b^*\ \mathtt{THEN\ RETURN}\ 1\ \mathtt{ELSE\ RETURN}\ 0.$

Figure D.1: Unforgeability and blindness for blind signatures

Definition D.2.5 [Partially Blind Signature Scheme] A *partially blind signature scheme* $\mathcal{PBS}$ is a 4-tuple of polynomial-time algorithms and protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$ generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- Signature Issuing: this is an interactive protocol between $\mathcal{S}(\mathsf{sk}, \mathsf{info})$ and $\mathcal{U}(\mathsf{vk}, m, \mathsf{info})$, for a message $m \in \mathcal{M}$ and shared information info. It generates an output σ for the user:
$$\sigma \leftarrow \langle \mathcal{S}(\mathsf{sk}, \mathsf{info}), \mathcal{U}(\mathsf{vk}, m, \mathsf{info}) \rangle \ .$$
- $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{info}, \sigma)$ outputs 1 if the signature σ is valid with respect to the message $m \| \mathsf{info}$ and vk, and 0 otherwise.

The security requirements are a direct extension of the classical ones: for unforgeability, we consider $m \| \mathsf{info}$ instead of m, and for blindness, we condition the unlinkability between signatures with the same public part info. Without the latter restriction, the signer could simply distinguish which message was signed by comparing the public information. The unforgeability is strengthened by considering also the public information so that the signer can be sure that the user will not be able to exploit his signature in another context.

Signer-Friendly Partially Blind Signatures. An agreement on info can be a long and tedious process allowing both participant to launch a denial-of-service attack. Instead of considering a global info, we will split it into two parts $\mathsf{info}_c, \mathsf{info}_s$, one chosen by the user and one by the signer. While we dismiss the agreement part, we stress that a signer can refuse to sign a public information he does not like, and a user can refuse to use a signature on a public information he did not agree with, so this division does not weaken the requirement on the scheme.

$\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{PBS},\mathcal{S}^*}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk}, m_0, m_1, st_{\texttt{FIND}}, \mathsf{info}_c, \mathsf{info}_s) \leftarrow \mathcal{S}^*(\texttt{FIND}, 1^\lambda)$;
$b \leftarrow \{0,1\}$;
$st_{\texttt{ISSUE}} \leftarrow \mathcal{S}^*(\texttt{ISSUE}, st_{\texttt{FIND}} : \langle \cdot, \mathcal{U}(\mathsf{vk}, m_b, \mathsf{info}_c)\rangle^{\leq 1}, \langle \cdot, \mathcal{U}(\mathsf{vk}, m_{1-b}, \mathsf{info}_c)\rangle^{\leq 1})$;
$\texttt{IF } \sigma_0 = \bot \texttt{ OR } \sigma_1 = \bot, (\sigma_0, \sigma_1) \leftarrow (\bot, \bot)$;
$b^* \leftarrow \mathcal{S}^*(\texttt{GUESS}, \sigma_0, \sigma_1, st_{\texttt{ISSUE}})$;
$\texttt{IF } b = b^* \texttt{ THEN RETURN } 1 \texttt{ ELSE RETURN } 0$.

Figure D.2: Blindness for Signer-Friendly Partially Blind Signatures

Definition D.2.6 [Signer-Friendly Partially Blind Signature Scheme] A *signer-friendly partially blind signature scheme* $\mathcal{PBS}$ is a 4-tuple of polynomial-time algorithms and protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \langle \mathcal{S}, \mathcal{U}\rangle, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$ generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- Signature Issuing is an interactive protocol between $\mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s)$ and $\mathcal{U}(\mathsf{vk}, m, \mathsf{info}_c)$, for a message $m \in \mathcal{M}$, signer information info_s and common information info_c. It generates an output σ for the user: $\sigma \leftarrow \langle \mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s), \mathcal{U}(\mathsf{vk}, m, \mathsf{info}_c)\rangle$.
- $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{info}_c, \mathsf{info}_s, \sigma)$ outputs 1 if the signature σ is valid with respect to the message $m \| \mathsf{info}_c \| \mathsf{info}_s$ and vk, and 0 otherwise.

We note that setting $\mathsf{info}_c := \mathsf{info}$ and $\mathsf{info}_s := \bot$ leads to a standard partially blind signature; whereas setting $\mathsf{info}_c = \mathsf{info}_s = \bot$ is the case of a standard blind signature. The signer always performs the last action in the signing protocol, and so if he does not want to sign a specific info, he can simply abort the protocol several times until the shared part suits his will, hence the name *signer-friendly*. This is why in the following we simply let him choose this input. If the user wants a specific word in the final message he can always add it to the blinded message. Intuitively this strengthens the unforgeability notion as the adversary (the user in this case) will not be able to choose the whole messages to be signed because of info_s. This is ensured in the security game, because the adversary must output valid signatures, therefore they should be done with the chosen info_s. For the blindness property, the adversary must choose two messages with the same public $\mathsf{info}_c \| \mathsf{info}_s$ component.

Security Games for Signer-Friendly Partially Blind Signatures. $\mathcal{PBS}$ satisfies *blindness* if, for any polynomial adversary $\mathcal{S}^*$ (malicious signer), the advantage $\mathsf{Succ}^{\mathsf{bl}\text{-}b}_{\mathcal{PBS},\mathcal{S}^*}(\lambda)$ is negligible, where

$$\mathsf{Succ}^{\mathsf{bl}}_{\mathcal{PBS},\mathcal{S}^*}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{bl}}_{\mathcal{PBS},\mathcal{S}^*}(\lambda) = 1] - 1/2| \ ,$$

in the security game presented in Figure D.2. If $\mathcal{S}^*$ refuses to sign one of the inputs (*i.e.* $\sigma_i = \bot$) then the two resulting signatures are set to $\bot$, therefore $\mathcal{S}^*$ does not gain any advantage if he decides to prevent the game execution. We let $\mathcal{S}^*$ choose both pieces of the public information, which corresponds to the case where, in the real world, the signer aborts as long as the user's public information does not suit him. As with regular partially blind signatures, the public information must be the same in both challenge messages to avoid a trivial attack. $\mathcal{PBS}$ is

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda)$
$(\mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
$\left((m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}, \sigma_i)\right)_{i\in\{1,\ldots,q_s+1\}} \leftarrow \mathcal{U}^*(\mathsf{vk} : \langle \mathcal{S}(\mathsf{sk},\cdot),\cdot\rangle^{\leq q_s})$;
`IF` $\exists i \neq j : (m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}) = (m_j, \mathsf{info}_{c,j}, \mathsf{info}_{s,j})$
`OR` $\exists i : \mathsf{Verify}(\mathsf{vk}, m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}, \sigma_i) = 0$ `THEN RETURN 0`
`ELSE RETURN 1`

Figure D.3: Unforgeability for Signer-Friendly Partially Blind Signatures

unforgeable if, for any polynomial adversary $\mathcal{U}^*$ (malicious user), the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda)$ is negligible, where

$$\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda) = 1] \ ,$$

in the security game presented in Figure D.3.

D.2.3 Efficient Instantiations of the Building Blocks

First, let us briefly sketch the basic building blocks: Groth-Sahai commitments, and a variation of the Waters signature. They both need a pairing-friendly environment $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible, non-degenerated, bilinear map, for two groups $\mathbb{G}$ and $\mathbb{G}_T$, of prime order p, generated by g and $g_t = e(g,g)$ respectively. From the following descriptions, it is easily seen that both schemes are randomizable.

Groth-Sahai Commitments. In 2008, Groth and Sahai [GS08] proposed non-interactive zero-knowledge proofs of satisfiability of certain equations over bilinear groups. Using as witness group elements (and scalars) which satisfy the equation, the prover starts with making commitments on them. The commitment key is of the form $(\mathbf{u}_1 = (u_{1,1} = g^{x_1}, 1, g), \mathbf{u}_2 = (1, u_{2,2} = g^{x_2}, g), \mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})) \in (\mathbb{G}^3)^3$. Depending on the definition of $\mathbf{u}_3$, this commitment can be either perfectly hiding or perfectly binding.

- To commit a group element $X \in \mathbb{G}$, one chooses three random scalars $r_1, r_2, r_3 \in \mathbb{Z}_p$ and sets (where $\odot$ denotes component-wise multiplication)

$$\begin{aligned}
\mathcal{C}(X) &:= (1,1,X) \odot \mathbf{u}_1^{r_1} \odot \mathbf{u}_1^{r_2} \odot \mathbf{u}_3^{r_3} \\
&= (c_1 = u_{1,1}^{r_1} \cdot u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} \cdot u_{3,2}^{r_3}, c_3 = X \cdot g^{r_1+r_2} \cdot u_{3,3}^{r_3}) \\
&= (c_1 = g^{x_1 r_1} \cdot u_{3,1}^{r_3}, c_2 = g^{x_2 r_2} \cdot u_{3,2}^{r_3}, c_3 = X \cdot g^{r_1+r_2} \cdot u_{3,3}^{r_3})
\end{aligned}$$

- To commit a scalar $x \in \mathbb{Z}_p$, one chooses two random scalars $\gamma_1, \gamma_2 \in \mathbb{Z}_p$ and sets

$$\begin{aligned}
\mathcal{C}'(x) &:= (1,1,g)^x \odot \mathbf{u}_1^{\gamma_1} \odot \mathbf{u}_3^{x+\gamma_2} \\
&= (c'_1 = u_{3,1}^{x+\gamma_2} \cdot u_{1,1}^{\gamma_1}, c'_2 = u_{3,2}^{x+\gamma_2}, c'_3 = u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1}) \\
&= (c'_1 = u_{3,1}^{x+\gamma_2} \cdot g^{x_1\gamma_1}, c'_2 = u_{3,2}^{x+\gamma_2}, c'_3 = u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1}).
\end{aligned}$$

The idea is that with a regular initialization of the commitment parameters ($\mathbf{u}_3 = \mathbf{u}_1^\nu \odot \mathbf{u}_2^\mu$, for two random scalars $\nu, \mu \in \mathbb{Z}_p$), these commitments are perfectly binding. The committed group

elements can even be extracted if one knows x_1, x_2: $c_3/(c_1^{1/x_1} c_2^{1/x_2}) = X$, and $c'_3/(c'_1{}^{1/x_1} c'_2{}^{1/x_2}) = g^x$.

However, if $\mathbf{u}_3$ is defined as $\mathbf{u}_3 = \mathbf{u}_1^{\nu} \odot \mathbf{u}_2^{\mu} \odot (1, 1, g^{-1}) = (u_{3,1} = u_{1,1}^{\nu}, u_{3,2} = u_{2,2}^{\mu}, u_{3,3} = g^{\nu+\mu-1})$, for two random scalars $\nu, \mu \in \mathbb{Z}_p$, the commitments are perfectly hiding. In addition, the two parameter initializations are indistinguishable under the DLin assumption. This is thus a mixed commitment.

To prove satisfiability of an equation (which is the statement of the proof), a Groth-Sahai proof uses these commitments and shows that the committed values satisfy the equation. The proof consists again of group elements and is verified by a pairing equation derived from the statement. In the perfectly binding setting the proof is perfectly sound, whereas in the perfectly hiding setting the proof perfectly hides the used witness.

Waters Signature. The *Waters signature scheme* was formally described in [Wat05]. It was proved existentially unforgeable against chosen-message attacks under the CDH assumption.

- $\mathsf{Setup}(1^\lambda)$: The scheme is defined over a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. The parameters are a randomly chosen generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $(u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ defining the function $\mathcal{F}\colon \{0,1\}^k \to \mathbb{G}$, $\mathcal{F}(M) = u_0 \prod u_i^{M_i}$. We set $\mathsf{param} := (p, \mathbb{G}, \mathbb{G}_T, e, g, h, (u_0, \ldots, u_k))$;
- $\mathsf{KeyGen}(\mathsf{param})$: Choose a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines $\mathsf{vk} = Y = g^y$, and $\mathsf{sk} = Z = h^y$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$: To sign a message $M \in \{0,1\}^k$, choose $s \xleftarrow{\$} \mathbb{Z}_p$ and define $\sigma = (\sigma_1 = Z \cdot \mathcal{F}(M)^s, \sigma_2 = g^s)$;
- $\mathsf{Verify}(\mathsf{vk} = Y, M, \sigma)$: Check whether $e(g, \sigma_1) \stackrel{?}{=} e(Y, h) \cdot e(\mathcal{F}(M), \sigma_2)$.

We also use another useful result on the Waters signature (as used in [LOS+06]):

Property 1 (Randomizability) *The Waters signature scheme is randomizable: for a valid pair (M, σ), if we define $\sigma' = (\sigma_1 \cdot \mathcal{F}(M)^{s'}, \sigma_2 \cdot g^{s'})$ for a random scalar s' then σ' is a random signature on M.*

Proof: If the initial signature was generated with randomness s, the modified signature corresponds to the signature on M with random coins $s + s'$. Since this is perfectly random in group $\mathbb{Z}_p$, it leads to a random signature on M. ∎

Suffixed Waters Signatures. Instead of signing one message, we will sign, with some additional parameters, a concatenation of 3 messages using Waters signatures:

$$m = M||\mathsf{info}_c||\mathsf{info}_s = (M_1, \ldots, M_\ell, \mathsf{info}_1, \ldots, \mathsf{info}_{k-\ell}) \in \{0,1\}^k$$

D.3 Signatures and Mixed Commitments

In [BFPV11] we presented a general framework for building extractable signatures on randomizable ciphertexts. Once a user has sent a ciphertext of a message m, and received a signature on this ciphertext, the framework proposes an algorithm SigExt which allows to recover the signature on the plaintext m, when one knows the decryption key. This property has also been strengthened to *strong extractability* where the initial user can recover the signature without possessing the decryption key if he has kept the randomness used in the initial ciphertext.

To achieve perfect blindness, we need to discard encryption and use a perfectly hiding commitment instead. This is where strong extractability becomes interesting, as we want the user to be able to recover the signature on the plaintext even when no decryption key exists.

D.3.1 Signatures on Mixed Commitments

We now define a scheme of signatures on mixed commitments. Note that this generalizes the existing definition of signatures on ciphertexts from [BFPV11].

Definition D.3.1 [Signatures on Mixed Commitments] A *signature scheme on a mixed commitment* $\mathcal{SC}$ is a 7-tuple of polynomial-time algorithms

$$(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{CKeyGen}, \mathsf{Commit}, \mathsf{Sign}, \mathsf{Decommit}, \mathsf{Verify}) :$$

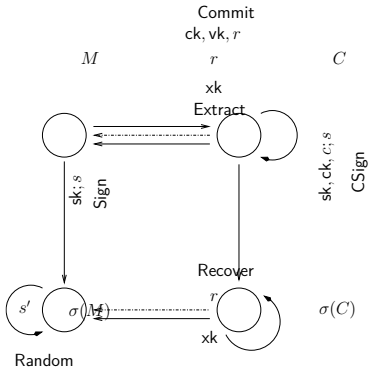
- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param for the associated encryption and signature schemes;
- $\mathsf{CKeyGen}(\mathsf{param})$ generates a commitment key ck and possibly the associated extraction key xk (according to the setting, *i.e.* whether $\mathsf{WISetup}$ or $\mathsf{ExSetup}$ was used);
- $\mathsf{SKeyGen}(\mathsf{param})$ generates a pair of keys, the verification key vk and the signing key sk;
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{vk}, m; r)$ produces a commitment c of $m \in \mathcal{M}$ under ck, using the random coins $r \in \mathcal{R}_c$. This commitment is intended to be later signed under the signing key associated to the verification key vk (the argument vk can be empty if the signing algorithm is universal and does not require a ciphertext specific to the signer);
- $\mathsf{CSign}(\mathsf{sk}, \mathsf{ck}, c; s)$, produces a signature σ on a commitment c and a signing key sk, using the random coins $s \in \mathcal{R}_s$, or $\perp$ if c is not valid (w.r.t. ck, and possibly vk associated to sk);
- $\mathsf{Decommit}(\mathsf{ck}, c, r, m)$ decommits c into a plaintext m, by showing that c is a valid commitment to m under ck with randomness r.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma)$ checks whether σ is a valid signature on c, w.r.t. the public key vk. It outputs 1 if the signature is valid, and 0 otherwise (possibly because of an invalid commitment c, with respect to ck, and possibly vk);
- $\mathsf{Recover}(\mathsf{vk}, \mathsf{ck}, c, \sigma, r)$ recovers a signature on the initial plaintext m committed in c, valid under vk using randomness r used in the commitment c.

One could have additionally defined an $\mathsf{Extract}$ algorithm to recover the signature on the initial plaintext, or even the initial plaintext itself from the commitment, using the extraction key. But the latter will not always exist, whereas the random coins used for the commitment always exist.

Strong security notions for signatures on ciphertexts are not meaningful in our context, as we want signatures to be efficiently malleable, as long as the plaintext is not affected. Hence we further weaken the original definition of *existential unforgeability* (EUF) [GMR88], where a new signature on an already signed message is not considered a forgery: In our definition a signature on a new ciphertext is not a forgery, if another ciphertext of the same plaintext has already been signed. Moreover, we require that the two setups for the mixed commitment be computationally indistinguishable. Under this assumption unforgeability in the perfectly binding setup also implies unforgeability in the perfectly hiding setting (in which we could not define the game).

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda)$
$(\mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$; $\mathcal{SM} := \emptyset$
$\{(\mathsf{ck}_i, \mathsf{xk}_i)\}_{i=1}^n \leftarrow \mathsf{CKeyGen}^n(\mathsf{param})$; $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$
$(c, \sigma) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{vk}, \mathsf{ck} : \mathsf{sign}(\mathsf{sk}, \cdot, \cdot))$;
$m \leftarrow \mathsf{Decommit}(\mathsf{xk}, \mathsf{vk}, c)$
`IF` $m = \perp$ `OR` $m \in \mathcal{SM}$ `OR` $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma) = 0$ `THEN RETURN` 0
`RETURN` 1

Figure D.4: Unforgeability of Signatures on Mixed Commitments



A message M can be committed using random coins r.
The signer can sign this ciphertext.
A signature on the plaintext can be obtained using either the coins r (Recover), or with the extraction key xk if we are in the perfectly binding setup; the result is the same as a signature of M by the signer (Sign). This final signature can be randomized.

Figure D.5: Recoverable Signatures on Mixed Commitments

We now define the unforgeability (UF) notion for signatures on mixed commitments, which makes sense in the perfectly binding setting only, but will help to prove some other security notions in any setting: $\mathcal{SC}$ is *unforgeable* if, for any polynomial-time adversary $\mathcal{A}$, the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda) = 1]$ is negligible, with $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}$ defined in Figure D.4. In this experiment $\mathsf{sign}(\mathsf{sk}, \cdot, \cdot)$ is an oracle that takes as input a previously generated commitment key ck_i and a commitment c, runs CSign on it, and returns its output σ. It also adds to the set $\mathcal{SM}$ of signed plaintexts the plaintext $m = \mathsf{Extract}(\mathsf{xk}_i, \mathsf{vk}, c)$.

Unforgeability in the above sense thus states that no adversary is able to generate a new valid commitment-signature pair for a commitment to new message, *i.e.* different to those contained in commitments that were queried to the signing oracle.

A signature on mixed commitments with recovery provides the following: a user can commit to a message m and obtain a signature σ on the commitment c. Knowing the randomness used to compute c, from (c, σ) one can not only recover the committed message m, but also a signature σ' on the *message* m, using the functionality Recover. The signature σ on the commitment c could thus be interpreted as a *commitment to a signature* on the message m. Commitment and signing can thus be seen as commutative (see Figure D.5). For completeness, we also define the Random algorithm for signatures, which we could also extend to signatures on committed values, but this is out of the scope of the applications this paper targets. Moreover, commitments must not be randomized if they are to be recovered later.

Figure D.5 makes sense for any setup of the mixed commitment, while the unforgeability game in Figure D.4 makes sense for the binding setup only.

Intuitively, the notion of blindness for our signatures comes from the hiding property of the commitment, because if the adversary can manage to find the order in which the message were signed then he has a bias in guessing which message is committed in each commitment (think of the final signature as a signature of a commitment). On the other hand, unforgeability of the

blind signature is implied by unforgeability of the underlying signature, as we can recover the signature under the extraction property (now the final signature is a commitment to a signature), which relies on the binding property of the commitment.

D.3.2 Our Construction

Our approach combines Groth-Sahai commitments [GS08] and the Waters signature [Wat05].

Assumptions. We rely on classical assumptions only: CDH for the unforgeability of signatures and DLin for the indistinguishability of the two commitment setups, which implies soundness of the proofs:

Definition D.3.2 [The Computational Diffie-Hellman problem (CDH)] The CDH assumption, in a cyclic group $\mathbb{G}$ of prime order p, states that for a generator $g \in \mathbb{G}$ and random $a, b \in \mathbb{Z}_p$, given (g, g^a, g^b), it is hard to compute g^{ab}.

Definition D.3.3 [Decision Linear Assumption (DLin)] The DLin assumption, in a cyclic group $\mathbb{G}$ of prime order p, states that given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$ for random $a, b, x, y \in \mathbb{Z}_p$, it is hard to determine whether $c = a + b$ or a random value. When $(g, u = g^x, v = g^y)$ is fixed, a tuple (u^a, v^b, g^{a+b}) is called a linear tuple w.r.t. (u, v, g), whereas a tuple (u^a, v^b, g^c) for a random and independent c is called a random tuple.

One can easily see that if an adversary is able to solve a CDH challenge, then he can solve a DLin one. So the DLin assumption implies the CDH assumption.

Scheme. Let us now describe our signature on Groth-Sahai commitments:

- $\mathsf{Setup}(1^\lambda)$ chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Moreover, it chooses an extra generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which defines the function $\mathcal{F}$. Setup returns $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F})$.
- $\mathsf{CKeyGen}(\mathsf{param})$ generates Groth-Sahai parameters $\mathsf{ck} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ in one of the two settings, and possibly the extraction key xk corresponding to the respective discrete logarithm in the binding setting.
- $\mathsf{SKeyGen}(\mathsf{param})$ chooses a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = Y = g^y$ and the secret key as $\mathsf{sk} = Z = h^y$.
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{vk}, m; r)$: For a message $M \in \{0,1\}^k$ and random scalars $r = (r_1, r_2, r_3) \xleftarrow{\$} \mathbb{Z}_p^3$, defines the commitment as

$$c = (c_1 = u_{1,1}^{r_1} u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} u_{3,2}^{r_3}, c_3 = g^{r_1+r_2} u_{3,3}^{r_3} \cdot \mathcal{F}(M))$$

  and computes $Y_{1,2} = Y^{r_1+r_2}, Y_3 = Y^{r_3}$. It moreover generates Groth-Sahai proofs of consistency of the commitment:

  - A proof Π_M of knowledge of M in c, which consists of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k)$ and proofs that each committed value is a bit, as well as a proof that c is a commitment to $\mathcal{F}(M)$. Π_M is composed of $9k + 3$ group elements.
  - A proof Π_r containing the commitments $C_r = (\mathcal{C}(Y_{1,2}), \mathcal{C}(Y_3))$ asserting that they are correctly generated. This requires 9 group elements.

  Π thus consists of $9k + 12$ group elements.

- $\mathsf{CSign}(\mathsf{sk}, \mathsf{ck}, (c, \Pi); s)$: To sign the commitment c, this first checks if the proof Π is valid. If so, it outputs $\sigma = (Z \cdot {c_3}^s, u_{3,3}^s, g^s)$, for a random scalar $s \in \mathbb{Z}_p$.
- $\mathsf{Decommit}(\mathsf{ck}, c, r, m)$: Decommits c to m by simply showing that a commitment to m using ck and r is equal to c.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma)$: Checks whether the following pairing equations are verified: $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(c_3, \sigma_3)$ and $e(\sigma_2, g) = e(u_{3,3}, \sigma_3)$.
- $\mathsf{Recover}(\mathsf{vk}, \mathsf{ck}, c, \sigma, r)$: If Verify is positive, one can use the randomness r to retrieve M and a valid signature on M:

$$\sigma' = (\sigma'_1 = \sigma_1 / (\sigma_3^{r_1+r_2} \sigma_2^{r_3}), \sigma'_2 = \sigma_3) \ ,$$

  which is a valid Waters signature.

Security Properties. In the next section, we generalize this construction to partially blind signatures and provide a security proof in any setting for the mixed commitment. Depending on the setting, we get either fair blind signatures (in the binding setting) or perfectly blind signatures when there is no shared public information.

Note that it suffices to do a security proof in one setting because of the indistinguishability of the two commitment setups. However, according to the setting, one will get perfect blindness or computational blindness.

D.4 Partially Blind Signatures

As in the previous section, our constructions will combine Groth-Sahai commitments [GS08] and Waters signatures [Wat05] as follows: given a commitment on the "Waters hash" $\mathcal{F}(M)$ (and some additional values proving knowledge of the message M and the randomness used) and a public shared information info_c, the signer can make a partially blind signature on M, info_c and an extra piece of public information info_s.

D.4.1 Partially Blind Signatures with Perfect Blindness

With those building blocks, we design a partially blind signature scheme, where the user sends a commitment to the message and gets back a signature on it by the signer. Thanks to the random coins of the commitment, the user can "unblind" the received Waters signature. Finally, by randomizing it, the user breaks all links between the message/signature pair and the transaction.

Our protocol proceeds as follows, on a commitment of $F = \mathcal{F}(M)$, a public common message info_c, and a public message info_s chosen by the signer. It is split into five steps, that correspond to an optimal 2-flow protocol: Blind, which is first run by the user, CSign, which is then run by the signer, and Verify, $\mathsf{Unblind}$, Random, which are successively run by the user to generate the final signature. We thus have $\mathcal{U} = (\mathsf{Blind}; \mathsf{Verify}, \mathsf{Unblind}, \mathsf{Random})$ and $\mathcal{S} = \mathsf{CSign}$:

- $\mathsf{Setup}(1^\lambda)$ first chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ and an additional generator $h \xleftarrow{\$} \mathbb{G}$. It generates a vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which, for messages $M \in \{0,1\}^\ell$ and with k being the overall length of $M||\mathsf{info}_c||\mathsf{info}_s$, defines

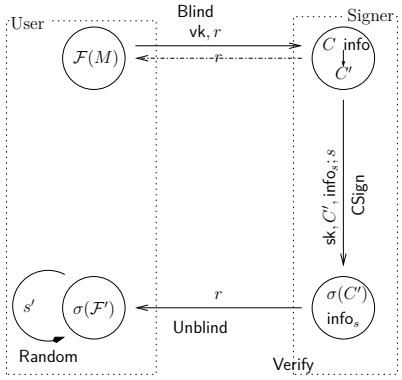
$$\mathcal{F}_1(M) = u_0 \prod_{i=1}^{\ell} u_i^{M_i} \qquad \mathcal{F}_2(\mathsf{info}) = \prod_{i=\ell+1}^{k} u_i^{\mathsf{info}_i}$$

  Moreover, it chooses Groth-Sahai parameters $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ in the perfectly hiding setting and outputs $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$.

- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = Y = g^y$ and the secret key as $\mathsf{sk} = Z = h^y$.
- Signature Issuing $\langle \mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s), \mathcal{U}(\mathsf{vk}, M, \mathsf{info}_c) \rangle$ is split into several steps:
  - $\mathsf{Blind}(M, \mathsf{vk}; (r_1, r_2, r_3))$: For a message $M \in \{0,1\}^\ell$ and random scalars $(r_1, r_2, r_3) \xleftarrow{\$} \mathbb{Z}_p$, defines the commitment as
    $$c = (c_1 = u_{1,1}^{r_1} u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} u_{3,2}^{r_3}, c_3 = g^{r_1+r_2} u_{3,3}^{r_3} \cdot \mathcal{F}(M))$$
    and computes $Y_{1,2} = Y^{r_1+r_2}, Y_3 = Y^{r_3}$. It also generates proofs of consistency:
    - A proof Π_M of knowledge of M in c, consisting of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_\ell))$, proofs that each committed value is a bit, and a proof that c commits to $\mathcal{F}(M)$. Π_M consists of $9\ell + 9$ group elements, as we have ℓ commitments, ℓ quadratic equations in $\mathbb{Z}_p$ of the type $M_i \cdot (1 - M_i) = 1$, and one quadratic multiscalar multiplication $u_0 \prod u_i^{M_i} = \mathcal{F}(M)$.
    - A proof Π_r containing the commitments $C_r = (\mathcal{C}(Y_{1,2}), \mathcal{C}(Y_3))$ asserting that they are consistent with c and C_M, *i.e.*, $e(c_3, Y) = e(g, Y_{1,2}) \cdot e(u_{3,3}, Y_3) \cdot e(\mathcal{F}(M), Y)$. This requires 9 additional group elements (6 for the commitments, and 3 for the Groth-Sahai proof).

    Π thus consists of $9\ell + 18$ group elements (when $\mathcal{M} = \{0,1\}^\ell$).
  - $\mathsf{CSign}(\mathsf{sk}, (c, \Pi), \mathsf{info}_c, \mathsf{info}_s; s)$: To sign the commitment c, first check if the proof Π is valid. Then append the public message $\mathsf{info} = \mathsf{info}_c \| \mathsf{info}_s$ to c_3 to create $c_3' = c_3 \cdot \mathcal{F}_2(\mathsf{info})$, which thus yields a commitment of the function evaluation on $M \| \mathsf{info}_c \| \mathsf{info}_s$ of global length k. Finally, output $\sigma = (Z \cdot (c_3')^s, u_{3,3}^s, g^s)$, for a random scalar $s \leftarrow \mathbb{Z}_p$, together with the additional public information info_s.
  - $\mathsf{Verify}(\mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), (\sigma_1, \sigma_2, \sigma_3))$: To check the validity of the signature, first compute $c_3' = c_3 \cdot \mathcal{F}_2(\mathsf{info})$ and then check whether the following pairing equations are satisfied:
    $$e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(c_3', \sigma_3) \qquad e(\sigma_2, g) = e(u_{3,3}, \sigma_3)$$
    If not then σ is not a valid signature, and the user sets the blind signature as $\Sigma = \perp$.
  - $\mathsf{Unblind}((r_1, r_2, r_3), \mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), \sigma)$: If the previous verification is successful, use the random coins r_1, r_2, r_3 to recover a Waters signature on $M \| \mathsf{info}_c \| \mathsf{info}_s$: $\sigma' = (\sigma_1' = \sigma_1 / (\sigma_3^{r_1+r_2} \sigma_2^{r_3}), \sigma_2' = \sigma_3)$.
  - $\mathsf{Random}(\mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), \sigma'; s')$: Randomize σ' to get the blind signature $\Sigma = (\sigma_1' \cdot \mathcal{F}(M || \mathsf{info}_c || \mathsf{info}_s)^{s'}, \sigma_2' \cdot g^{s'})$.

  Note that Σ is a random Waters signature on $M || \mathsf{info}_c || \mathsf{info}_s$, where we denote $F = \mathcal{F}(M || \mathsf{info}_c || \mathsf{info}_s)$:
  $$\begin{aligned}
  \Sigma &= (\sigma_1' \cdot F^{s'}, \sigma_2' \cdot g^{s'}) = (F^{s'} \cdot \sigma_1 / (\sigma_3^{r_1+r_2} \sigma_2^{r_3}), g^{s'} \cdot \sigma_3) \\
  &= (F^{s'} \cdot Z \cdot {c_3'}^s / (g^{s(r_1+r_2)} u_{3,3}^{sr_3}), g^{s+s'}) \\
  &= (F^{s'} \cdot Z \cdot g^{s(r_1+r_2)} u_{3,3}^{sr_3} \cdot F^s / (g^{s(r_1+r_2)} u_{3,3}^{sr_3}), g^{s+s'}) \\
  &= (F^{s+s'} \cdot Z, g^{s+s'}) \;=\; \mathsf{Sign}(Z, M || \mathsf{info}_c || \mathsf{info}_s; s + s')
  \end{aligned}$$
- $\mathsf{Verify}(\mathsf{vk}, (M, \mathsf{info}_c, \mathsf{info}_s), \Sigma = (\Sigma_1, \Sigma_2))$, is defined as Waters signature verification by checking that the following holds:
  $$e(\Sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M || \mathsf{info}_c || \mathsf{info}_s), \Sigma_2) \ .$$



Blind hides a message M using random coins r. The signer can concatenate public messages info_c and info_s to the original commitment, which yields a commitment C' on $F = \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)$.
Using randomness r, Unblind recovers a signature on the plaintext, which is the same as a direct signature on $M||\mathsf{info}_c||\mathsf{info}_s$ by the signer. Randomizing this signature prevents the signer to link the transaction to a signature.

Figure D.6: Partially Blind Signatures with Perfect Blindness

Theorem D.4.1 This signer-friendly partially blind signature scheme is unforgeable under the DLin assumption in $\mathbb{G}$.

Proof: Let us denote $\mathcal{PBS}$ our partially blind signature above. Assuming there is an adversary $\mathcal{A}$ against unforgeability that succeeds with probability ϵ, we build an adversary $\mathcal{B}$ against CDH, whose hardness is implied by that of DLin.

DLin Assumption. Breaking unforgeability means that after q_s interactions with the signer, the adversary manages to output $q_s + 1$ valid message/signature pairs on distinct messages. If the adversary $\mathcal{A}$ succeeds with probability ϵ when the commitment key is perfectly hiding, then $\mathcal{A}$ also succeeds with a probability negligibly close to ϵ when the commitment key is perfectly hiding. Otherwise $\mathcal{A}$ could be used to distinguish the two types of commitment keys, which are indistinguishable under DLin.

Signer simulation. Let us thus consider $\mathcal{PBS}$ but with a commitment scheme using the *binding* setting, say $\mathcal{PBS}'$. Our simulator $\mathcal{B}$ can thus extract values from the commitments since it will know the extraction key. We assume that $\mathcal{A}$ is able to break the unforgeability of $\mathcal{PBS}'$ with probability ϵ' after q_s interactions with the signer, based on which we build an adversary $\mathcal{B}$ against the CDH problem. (We basically follow Waters' original proof, but adapt it, since we need to simulate signatures $(\sigma_1, \sigma_2, \sigma_3)$ on commitments, rather than plain Waters signatures.)

Let $(A = g^a, B = g^b)$ be a challenge CDH-instance in a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. We generate the global parameters using this instance: for simulating Setup/KeyGen, $\mathcal{B}$ picks a random position $j \xleftarrow{\$} \{0, \ldots, k\}$, chooses random indices $y_0, y_1, \ldots, y_k \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$, and random scalars $z_0, z_1, \ldots, z_k \xleftarrow{\$} \mathbb{Z}_p$. Define $Y = A = g^a$, $h = B = g^b$, $u_0 = h^{y_0 - 2jq_s} g^{z_0}$, and $u_i = h^{y_i} g^{z_i}$ for $i = 1, \ldots, k$. $\mathcal{B}$ also picks random scalars ν, μ, x_1, x_2, and generates binding Groth-Sahai parameters $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ with $(\mathbf{u}_1 = (u_{1,1} = g^{x_1}, 1, g), \mathbf{u}_2 = (1, u_{2,2} = g^{x_2}, g), \mathbf{u}_3 = \mathbf{u}_1^{\nu} \odot \mathbf{u}_2^{\mu})$. Note that $u_{3,3} = g^{\nu+\mu}$. It outputs $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$. (The signing key is implicitly defined as $Z = h^a = B^a = g^{ab}$, which is the solution to our Diffie-Hellman instance.)

To answer a signing query for $(c = (c_1, c_2, c_3), \Pi)$, $\mathcal{B}$ first checks the proof $\Pi = (\Pi_M, \Pi_r)$. Using the commitment extraction key (x_1, x_2), it extracts M from the bit-by-bit commitments in Π_M and $Y_{1,2} = Y^{r_1+r_2}$, $Y_3 = Y^{r_3}$ from Π_r (where r_1, r_2, r_3 are the random coins of c). Furthermore, we can compute $c'_3 = g^{r_1+r_2} u_{3,3}^{r_3} \cdot F$, where we denote $M' = M||\mathsf{info}_c||\mathsf{info}_s$ and $F = \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)$. $\mathcal{B}$ defines

$$H = -2jq_s + y_0 + \textstyle\sum_i y_i M'_i \qquad J = z_0 + \textstyle\sum_i z_i M'_i \qquad F = h^H g^J$$

If $H \equiv 0 \pmod{p}$ then $\mathcal{B}$ aborts, otherwise it sets

$$\sigma = (Y^{-J/H}(Y_{1,2}Y_3^{\nu+\mu})^{-1/H}(F(c_1^{1/x_1}c_2^{1/x_2}))^s, (Y^{-1/H}g^s)^{\nu+\mu}, Y^{-1/H}g^s).$$

Defining $\tilde{s} = s - a/H$, we have

$$\begin{aligned}\sigma_1 =& Y^{-J/H}(Y_{1,2}Y_3^{\nu+\mu})^{-1/H}(h^H g^J(c_1^{1/x_1}c_2^{1/x_2}))^s = Z \cdot (c_3')^{\tilde{s}}\\ \sigma_3 =& Y^{-1/H}g^s = Y^{-1/H}g^{\tilde{s}+a/H} = g^{\tilde{s}}\\ \sigma_2 =& (\sigma_3)^{\nu+\mu} = g^{(\nu+\mu)\tilde{s}} = u_{3,3}^{\tilde{s}}\end{aligned}$$

This thus exactly looks like a real signature sent by the signer.

Diffie-Hellman extraction. After at most q_s signing queries $\mathcal{A}$ outputs $q_s + 1$ valid Waters signatures. Since these are more than the number of signing queries, there is a least one message M^* that is different from all the messages $M||\mathsf{info}_c||\mathsf{info}_s$ involved in the signing queries. We define

$$H^* = -2jq_s + y_0 + \textstyle\sum_i y_i M_i^*, \qquad J^* = z_0 + \textstyle\sum_i z_i M_i^* \qquad \mathcal{F}(M^*) = h^{H^*}g^{J^*}$$

If $H^* \not\equiv 0 \pmod{p}$ then $\mathcal{B}$ aborts, otherwise, for some s^*, we have $\sigma^* = (h^a\mathcal{F}(M^*)^{s^*}, g^{s^*}) = (h^a g^{s^* J^*}, g^{s^*})$. Then, $\sigma_1^*/(\sigma_2^*)^{J^*} = h^a = g^{ab}$, which is a solution for the CDH problem.

Success probability. (this is based on [HK08]) The Waters hash function is $(1, q_s)$-programmable (*i.e.*, we can find with non negligible probability a case where q_s intermediate hashes H are non zero, and the last one is); therefore the previous simulation succeeds with non negligible probability ($\Theta(\epsilon/q_s\sqrt{k})$), with which $\mathcal{B}$ then breaks CDH. ∎

Theorem D.4.2 This signer-friendly partially blind signature scheme achieves perfect blindness.

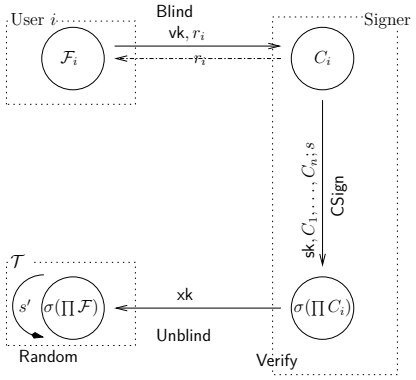
Proof: Since the commitment key is perfectly hiding, the transcript sent to the signer contains a commitment on the message to be signed which leaks no information about M. The additional proofs are perfectly witness-indistinguishable and thus do not provide any additional information about M. (For Groth-Sahai proofs in the perfectly hiding setting, for any M, committed with randomness r and any message M', there exists a random r' such that both commitment values collide.) Moreover, due to the perfect randomizability of Waters signatures, the output blind signature is uniformly random in the set of signatures on $M||\mathsf{info}_c||\mathsf{info}_s$, on which no information leaked. So the resulting signature is independent from the transcript seen by the signer. ∎

D.5 Multi-Source Blind Signatures

D.5.1 Concatenation

The previous constructions enables a user to obtain a signature on a plaintext without revealing it to the signer. But what if the original message is coming from various users? We now present a new way to obtain a blind signature without requiring multiple users to combine their messages, providing once again a round-optimal way to achieve our goal.

We thus consider another variant of our blind signature scheme. Setup no longer creates perfectly hiding parameters, but perfectly binding parameters. We therefore need not compute $u_{3,3}^s$ to run Unblind, since we can use the extraction key instead of the coins. In addition, in this



Different users can hide messages M_i using random coins r_i (Blind).
The signer can concatenate the messages inside the commitments, creating a commitment on $\mathcal{F} = \prod \mathcal{F}_i$.
Using the extraction key xk in Unblind, the tallier $\mathcal{T}$ can recover a signature on the plaintext, which is the same as a signature by the signer on the concatenation of all the messages M_i.
Randomizing this signature prevents the signer from linking the signature to a transaction.

Figure D.7: Multi-Source Blind Signature on Concatenation

scenario we do not consider a unique user providing a blinded message, but several users. The signer will produce a signature on a multi-source message, provided as different commitments. The signature and the messages will actually be committed under a key from a third party, which will be the only one able to extract the message and the signature.

Our instantiation is similar to the previous ones in the perfectly binding setting. For simplicity, we remove the partially blind part, but of course it could be adapted in the same way.

With the previous building blocks, we will sign several commitments of $F_i = \mathcal{F}_i(M_i)$, and instead of the protocol $\langle \mathcal{U}, \mathcal{S} \rangle$, we now have one with three kind of participants: users $\mathcal{U}_i$ will blind a commitments on $\mathcal{F}_i(M_i)$, signer $\mathcal{S}$ signs the blinded message, and $\mathcal{T}$, the tallier, will verify, unblind and randomize this signature:

- $\mathsf{Setup}(1^\lambda)$: In a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ the algorithm outputs a generator $h \xleftarrow{\$} \mathbb{G}$ and a vector
$$\vec{u} = (u_0, (u_{i,1}, \ldots, u_{i,\ell})_{i=1}^n) \xleftarrow{\$} \mathbb{G}^{n\ell+1} \ ,$$
where ℓ is a polynomial in λ. We define $\mathcal{F}_i(M_i) = \prod_\ell u_{i,\ell}^{m_{i,\ell}}$.

- $\mathsf{KeyGen}(\mathsf{param})$: Choose $y \xleftarrow{\$} \mathbb{Z}_p$, which defines $\mathsf{vk} = Y = g^y$ and $\mathsf{sk} = Z = h^y$, and generate a perfectly binding Groth-Sahai commitment key ck together with an extraction key $\mathsf{xk} = (x_1, x_2) \in \mathbb{Z}_p^2$.

- $(\mathcal{U}_i, \mathcal{S}, \mathcal{T})$:
  - $\mathsf{Blind}(M, \mathsf{vk}; (r_1, r_2, r_3))$ (we omit the subscripts i): For a message $M \in \{0,1\}^\ell$ and random scalars in $\mathbb{Z}_p$, define the commitment $c = \mathcal{C}(\mathcal{F}(M)) = (c_1, c_2, c_3)$. As before, we add proofs to this commitment:
    - A proof Π_M of knowledge of M, such that c commits to $\mathcal{F}(M)$, which consists of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$ and proofs that each committed value is a bit.
    - A proof Π_r containing the commitments $(\mathcal{C}(Y^{r_1+r_2}), \mathcal{C}(Y^{r_3}))$ together with proofs of consistency with c and C_M.
  - $\mathsf{CSign}(\mathsf{sk}, (c = (c_{1,i}, c_{2,i}, c_{3,i}), \Pi_i)_{i=1}^n; s)$: To sign n commitments, first check that all Π_i's are valid; after randomizing these commitments, compute the global commitment $C = (\prod c_{1,i}, \prod c_{2,i}, u_0 \prod c_{3,i})$ and output $C = (C_1, C_2, C_3)$ and $\sigma = (C_1^s, C_2^s, Z \cdot C_3^s, g^s)$.
  - $\mathsf{Verify}(\mathsf{vk}, (C_1, C_2, C_3), (\sigma_1, \sigma_2, \sigma_3, \sigma_4))$: In order to check validity of the signature, verify the following equations: $e(\sigma_1, g) = e(C_1, \sigma_4)$, $e(\sigma_2, g) = e(C_2, \sigma_4)$, and $e(\sigma_3, g) = e(h, \mathsf{vk}) \cdot e(C_3, \sigma_4)$.

  - $\mathsf{Unblind}(\mathsf{xk}, \mathsf{vk}, c = (C_1, C_2, C_3), \Pi = (\Pi_i)_{i=1}^n, \sigma)$: From a valid signature σ on c, knowing the extraction key (x_1, x_2), one can extract the message M from the bit-by-bit commitments in Π. One can also extract the corresponding signature σ' as: $\sigma'_1 = \sigma_3/(\sigma_1^{1/x_1}\sigma_2^{1/x_2}), \sigma'_2 = \sigma_4$, which is a Waters signature on M, the concatenation of all the messages.
  - $\mathsf{Random}(\mathsf{vk}, M, \sigma'; s')$: The signature σ' is randomized to get the blind signature $\Sigma = (\sigma'_1 \cdot \mathcal{F}(M)^{s'}, \sigma'_2 \cdot g^{s'})$.

- $\mathsf{Verify}(\mathsf{vk}, M, \sigma = (\sigma_1, \sigma_2))$: In order to check the validity of the signature, one checks whether: $e(\sigma_1, g) \stackrel{?}{=} e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

Theorem D.5.1 The above multi-source blind signature scheme for concatenation is blind and unforgeable under the DLin and CDH assumptions, respectively: no adversary can generate more message/signature pairs on distinct messages than the number of interactions with the signer.

The theorem follows from the previous result, as the combination of the different partial Waters hashes can be seen as one global hash (note that we have independent generators for each index). The perfectly binding Groth-Sahai proofs guarantee that each user outputs a Waters hash of their message under their generators.

D.5.2 Linear Operations

The previous scheme presents a way to combine multiple blind messages into one by concatenation. One drawback is that every bit in every one of the concatenated messages requires a generator in the setup. Let us now assume all that is required is a signature on the sum of the messages (or the mean, or any other linear operation). Concatenation could still be used, as the verifier could perform the linear operation at the end, but all individual messages are signed, requiring a long public key. We improve the construction, reducing the public key size and the information leaked about the individual messages when one only requires a signature on the sum or the mean of the individual messages. Instead of signing the concatenation of the messages, we now consider the sum of messages, for which we can allow every user to use the same generators for the function $\mathcal{F}$, which reduces the public key by a factor of the number of aggregated messages.

The resulting scheme is the same as before except that for Setup we have $\vec{u} = (u_0, \ldots, u_k) \stackrel{\$}{\leftarrow} \mathbb{G}^{k+1}$. We then proceed as before considering $\mathcal{F}(M_i) = \prod_\ell u_\ell^{m_{i,\ell}}$, which are aggregated to $\prod_i \mathcal{F}(M_i)$, which is $\mathcal{F}$ evaluated on the *sum* of all messages. However, the exponents in the Waters hash function are no longer bits but belong to a larger alphabet (*e.g.* $\{0, \ldots, t\}$ if t users participate and send bit strings). Following the work done in [HK08], we will show in the next section that over a non-binary alphabet the Waters function remains $(1, poly)$-programmable as long as the size of the alphabet is a polynomial in the security parameter. This result readily implies the security of the multi-source blind signature scheme for addition, but also any linear combination.

Theorem D.5.2 This multi-source blind signature scheme for addition is blind and unforgeable under the DLin assumption as long the alphabet size and the number of sources are polynomial in the security parameter.

Note however that more than just the sum is leaked since carries are not propagated, but accumulated in each digit. Hence, the value of the sum in one digit leaks some information on the same digits on individual messages. Anyway, our goal is to authenticate the result with minimal communication.

D.6 Waters Function and Non-binary Alphabets

In this section we prove that for a polynomial-size alphabet, the Waters function remains programmable. We recall some notations introduced in [HK08] and show our result, which can be seen as an improvement over that presented by Naccache [Nac07], who considered a variant of Waters identity-based encryption [Wat05] with shorter public parameters.

D.6.1 Definitions

Let us recall some basic definitions. Considering a cyclic group $\mathbb{G}$, for some security parameter λ, we define a *group hash function* H for G, an alphabet $\Sigma = \Sigma(\lambda)$ and an input length $\ell = \ell(\lambda)$ as a pair of probabilistic polynomial-time algorithms $(\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ such that:

- $\mathsf{PHF.Gen}$ takes as input a security parameter λ and outputs a key κ
- $\mathsf{PHF.Eval}$ takes as input a key κ output by $\mathsf{PHF.Gen}$ and a string $X \in \Sigma^\ell$ and outputs an element of $\mathbb{G}_\lambda$.

Definition D.6.1 [[HK08]] A group hash function $(\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ is (m, n, δ)-programmable, if there exist two PPT algorithms $(\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$ such that:

- **Syntax:** For $g, h \in \mathbb{G}$, $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ generates a key κ' and a trapdoor t such that $\mathsf{PHF.TrapEval}(t, X)$ produces integers a_X, b_X for any $X \in \Sigma^\ell$.
- **Correctness:** For all generators $g, h \in \mathbb{G}$, all $(\kappa', t) \leftarrow \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ and all string $X \in \Sigma^\ell$, $H_{\kappa'}(X) := \mathsf{PHF.Eval}(\kappa', X)$ satisfies $H_{\kappa'}(X) = g^{a_X} h^{b_X}$ where $(a_X, b_X) := \mathsf{PHF.TrapEval}(t, X)$.
- **Statistically close trapdoor keys:** For all generators $g, h \in \mathbb{G}$, the two algorithms $\mathsf{PHF.Gen}(1^\lambda)$ and $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ output keys κ and κ' statistically close.
- **Well-distributed logarithms:** For all generators $g, h \in \mathbb{G}$, all keys (κ', t) output by $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ and all strings $(X_i)_{1,\ldots,m}, (Z_i)_{1,\ldots,n} \in \Sigma^\ell$ such that $\forall i, j, X_i \neq Z_j$, we have $\Pr[a_{X_1} = \ldots, a_{X_m} = 0 \wedge a_{Z_1} \cdot \ldots \cdot a_{Z_n} \neq 0] \geq \delta$, where the probability is taken over the random coins used by $\mathsf{PHF.TrapGen}$ and $(a_{X_i}, b_{X_i}) := \mathsf{PHF.TrapEval}(t, X_i)$ and $(a_{Z_i}, b_{Z_i}) := \mathsf{PHF.TrapEval}(t, Z_i)$.

D.6.2 Instantiation with Waters Function

Let us consider the Waters function presented in [Wat05].

Definition D.6.2 [Multi-Generator PHF] Let $G = (\mathbb{G}_\lambda)$ be a group family, and $\ell = \ell(\lambda)$ a polynomial. We define $\mathcal{F} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ as the following group hash function:

- $\mathsf{PHF.Gen}(1^\lambda)$ outputs $\kappa = (h_0, \ldots, h_\ell) \xleftarrow{\$} \mathbb{G}^{\ell+1}$;
- $\mathsf{PHF.Eval}(\kappa, X)$ parses κ and $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ and then outputs $\mathcal{F}_\kappa(X) = h_0 \prod_{i=1}^{\ell} h_i^{x_i}$.

This function was shown to be $(1, q, \delta)$-programmable with $\delta = O(1/(q\sqrt{\ell}))$ and $(2, 1, \delta)$-programmable with $\delta = O(1/\ell)$ (*cf.* [HK08]). However, this definition requires to generate and store $\ell + 1$ group generators where ℓ is the bit-length of the messages one wants to hash. We consider a more general case where instead of hashing bit-per-bit we hash blocks of bits.

Definition D.6.3 [Improved Multi-Generator PHF] Let $G = (\mathbb{G}_\lambda)$ be a group family, $\Sigma = \{0, \ldots, \tau\}$ a finite alphabet and $\ell = \ell(\lambda)$ a polynomial. We define $\mathcal{F} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ as the following group hash function:

- $\mathsf{PHF.Gen}(1^\lambda)$ returns $\kappa = (h_0, \ldots, h_\ell) \xleftarrow{\$} \mathbb{G}^{\ell+1}$;
- $\mathsf{PHF.Eval}(\kappa, X)$ parses $\kappa = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$ and $X = (x_1, \ldots, x_\ell) \in \Sigma^\ell$ and then outputs ${\mathcal{F}^+}_\kappa(X) = h_0 \prod_{i=1}^{\ell} h_i^{x_i}$.

Using a larger alphabet allows to hash from a larger domain with a smaller hash key, but it comes at a price: we show that the function is no longer $(2,1)$-programmable (*i.e.*, no longer $(2,1,\delta)$ programmable for a non-negligible δ).

Theorem D.6.4 [(2,1)-Programmability] For any group family G with known order and $\tau > 1$, the function $\mathcal{F}^+$ is not a (2,1)-programmable hash function if the discrete logarithm problem is hard in G.

Proof: Consider a discrete logarithm challenge (g, h) in a group $\mathbb{G}_\lambda$ and suppose by contradiction that the function $\mathcal{F}^+$ is $(2,1)$-programmable with $\tau \geq 2$ (*i.e.*, we suppose that there exist two probabilistic polynomial-time algorithms $(\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$ satisfying Definition D.6.1 for a non-negligible δ).

For any hash key κ' and trapdoor t generated by $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$, we can consider the messages $X_1 = (2,0), X_2 = (1,1), Z = (0,2)$.

With non-negligible probability over the random coins used by $\mathsf{PHF.TrapGen}$ we have $a_{X_1} = a_{X_2} = 0$ and $a_Z \neq 0$ where $(a_{X_1}, b_{X_1}) := \mathsf{PHF.TrapEval}(t, X_1)$, $(a_{X_2}, b_{X_2}) := \mathsf{PHF.TrapEval}(t, X_2)$ and $(a_Z, b_Z) := \mathsf{PHF.TrapEval}(t, Z)$. By the correctness property, we have

$$g^{a_Z} h^{b_Z} = \mathcal{F}(Z) = h_0 h_2^2 = (\mathcal{F}(X_2))^2 / \mathcal{F}(X_1) = h^{2b_{X_2}} / h^{b_{X_1}}$$

and we can extract the discrete logarithm of g in base h as follows:

$$\log_h(g) = \frac{2b_{X_2} - b_{X_1} - b_Z}{a_Z} \mod |\mathbb{G}_\lambda| \ .$$

▌ However we still have the following interesting property:

Theorem D.6.5 [(1,poly)-Programmability] For any polynomial q and a group family G with groups of known order, the function $\mathcal{F}^+$ is a $(1, q, \delta)$-programmable hash function with a $\delta = \Omega(1/\tau q\sqrt{\ell})$.

Remark D.6.6 This theorem improves the result presented by Naccache in [Nac07] where the lower bound on the $(1, q, \delta)$-programmability was only $\delta = \Omega(1/\tau q \ell)$.

Remark D.6.7 In order to be able to sign all messages in a set $\mathcal{M}$, we have to consider parameters τ and ℓ such that $\tau^\ell \geq \#\mathcal{M}$, but the security is proved only if the value δ is non-negligible (*i.e.* if $\ell = \lambda^{O(1)}$ and $\tau = \lambda^{O(1)}$). In particular if $\mathcal{M}$ is of polynomial size in λ (which is the case in our WSN application with data aggregation), one can use $\tau = \#\mathcal{M}$ and $\ell = 1$ (namely, the Boneh-Boyen hash function [BB04a]), and therefore get data confidentiality.

Proof: Let us first introduce some notation. Let $n \in \mathbb{N}^*$; for $j \in \{1, \ldots, n\}$, let A_j be independent and uniform random variables in $\{-1, 0, 1\}$. If we denote $2\sigma_j^2$ their quadratic moment, we have $2\sigma_j^2 = 2/3$ and $\sigma_j = \sqrt{1/3}$. We note $s_n^2 = \sum_{j=1}^n \sigma_j^2 = n/3$.

The Local Central Limit Theorem. Our analysis relies on a classical result on random walks, called the *Local Central Limit Theorem*[1] . It basically provides an approximation of $\Pr[\sum A_j = a]$ for independent random variables A_j. This is a version of the Central Limit Theorem in which the conclusion is strengthened from convergence of the law to locally uniform pointwise convergence of the densities. It is worded as follows in [DM95, *Theorem 1.1*], where ϕ and Φ are the standard normal density and distribution functions:

Theorem D.6.8 Let A_j be independent, integer-valued random variables where A_j has probability mass function f_j (for $j \in \mathbb{N}^*$). For each $j, n \in \mathbb{N}^*$, let $q(f_j) = \sum_k \min(f_j(k), f_j(k+1))$ and $Q_n = \sum_{j=1}^n q(f_j)$. Denote $S_n = A_1 + \cdots + A_n$. Suppose that there are sequences of numbers $(\alpha_n), (\beta_n)$ such that

1. $\lim_{n\to\infty} \Pr[(S_n - \alpha_n)/\beta_n) < t] = \Phi(t), -\infty < t < \infty$,
2. $\beta_n \to \infty$,
3. and $\limsup \beta_n^2/Q_n < \infty$,

then[2] $\sup_k |\beta_n \Pr[S_n = k] - \phi((k - \alpha_n)/\beta_n)| \to 0$ as $n \to \infty$.

While those notations may seem a little overwhelming, this can be easily explained in our case. With $A_j \in \{-1, 0, 1\}$ with probability $1/3$ for each value.

1. It requires the variables to verify the Lindeberg-Feller theorem. However as long as the variables verify Lindeberg's condition[3], this is true for $\beta_n = s_n$ and $\alpha_n = 0$.
2. In our application, $\beta_n = s_n = \sqrt{n/3}$, so again we comply with the condition.
3. Since $f_j(k)$ is simply the probability that A_j equals k, then $q(f_j) = 2/3$. This leads to $Q_n = 2n/3$. As a consequence, $\beta_n^2/Q_n = 1/2$.

So we have: $\sup_k |\beta_n \Pr[S_n = k] - \phi((k - \alpha_n)/\beta_n)| \to 0$, that is, in our case

$$\sup_k |\sqrt{n/3} \Pr[S_n = k] - \phi(k/\sqrt{n/3})| \to 0 \ .$$

We solely focus on the case $k = 0$: since $\phi(0) = 1/\sqrt{2\pi}$, $\Pr[S_n = 0] = \Theta(1/\sqrt{n})$. In addition, it is clear that $\Pr[S_n = k] \leq \Pr[S_n = 0]$ for any $k \neq 0$ (*cf.* [HK08]).

Lemma D.6.9 Let $(A_{ij})_{[\![1,n]\!]\times[\![1,J]\!]}$ be independent, integer-valued random variables in the set $\{-1, 0, 1\}$, then for all $X \in [\![1, \tau]\!]^n$, $\Pr[\sum_{i=1}^n \sum_{j=1}^J X_i A_{ij} = 0] = \Omega(1/\tau\sqrt{nJ})$, where the probability distribution is over the A_{ij}.

[1] The main idea here is to show that even if the probability that the studied random walks end in 0 is maximal, this probability in neither negligible nor overwhelming.

[2] The so-called Berry-Esseen theorem gives the rate of convergence of this supremum.

[3] Lindeberg's condition is a sufficient criteria of the Lindeberg-Feller theorem, for variables with a null expected value it requires that $\forall \epsilon > 0, \lim_{n\to\infty} 1/s_n^2 \sum_{j=1}^n E[A_j^2 \cdot 1_{\{|A_j| > \epsilon s_n\}}] \to 0$. In our case, as soon as $n > 3/\epsilon^2$, we have $|A_j| \leq 1 \leq \epsilon\sqrt{n/3} \leq \epsilon s_n$, so the sum is zero. ($1_{\{|A_j|>\epsilon s_n\}}$ is the indicator function of variables greater that ϵs_n)

This lemma will be useful to prove the lower bound in the following, we only consider word with no null coefficient X_i, if a X_i is null, we simply work with a shorter random walk of length $J \cdot (n-1)$ instead of Jn.

Proof: Let us denote d_{ij}, the random variable defined as $X_i A_{ij}$: they are independent, integer-valued random variables. As above, $s_n^2 = \sum_{i=1}^{n} \sum_{j=1}^{J} \sigma_j^2 = \sum_{i=1}^{n} JX_i^2/3$. So $nJ/3 \leq s_n^2 \leq n\tau^2 J/3$.

1. Lindeberg's condition is verified. As soon as $n > 3\tau/J\epsilon^2$ we have $\epsilon s_n > \tau$ and so $|d_{ij}| < s_n$, and so once again the sum is null.
2. $s_n \to \infty$.
3. Each $d_{ij} \in \{-X_i, 0, X_i\}$ with probability 1/3 for each value, so $q(f_{ij}) = 2/3$ and $Q_n = \sum_{i,j} q(f_{ij}) = 2nJ/3$. So $\beta_n^2/Q_n \leq (n\tau J/3)/(2nJ/3) \leq \tau/2 < \infty$.

Then we can apply the Local Central Limit Theorem to the d_{ij}'s, and conclude:

$$\Pr[\sum_{i=1}^{n} \sum_{j=1}^{J} X_i A_{ij} = 0] = \Theta(1/s_n) = \Theta(1/\tau\sqrt{nJ}).$$

∎

In the following, we will denote $a(X) = \sum_{i=1}^{n} a_i X_i$, where $X \in \{0, \ldots, \tau\}^n$. The probabilities will be over the a_{ij}'s variables while X and Y are assumed to be chosen by the adversary. Our goal is to show that even for bad choices of X and Y, a random draw of a_{ij}'s provides enough freedom.

Let $J = J(\lambda)$ be a positive function. We define the following two probabilistic polynomial-time algorithms ($\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval}$):

- $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$**:** which picks independently and uniformly at random elements $(a_{ij})_{(0,\ldots,\ell),(1,\ldots,J)}$ in $\{-1, 0, 1\}$, and random exponents $(b_i)_{(0,\ldots,\ell)}$. It sets $a_i = \sum_{j=1}^{J} a_{ij}$ and $h_i = g^{a_i} h^{b_i}$ for $i \in \{0, \ldots, \ell\}$. It then outputs the hash key $\kappa = (h_0, \ldots, h_\ell)$ and the trapdoor $t = (a_0, b_0, \ldots, a_\ell, b_\ell)$.
- $\mathsf{PHF.TrapEval}(t, X)$**:** which parses $X = (X_1, \ldots, X_\ell) \in \Sigma^\ell = \{0, \ldots, \tau\}^\ell$ and outputs $a_X = a_0 + \sum a_i X_i$ and $b_X = b_0 + \sum b_i X_i$.

As this definition verifies readily the syntactic and correctness requirements, we only have to prove the two other ones. We stress the importance of the hardwired 1 in front of a_0 this allows us to consider multisets $X' = 1 :: X$ and $Y' = 1 :: Y$, and so there is no k such that $X' = kY'$. And we also stress that $a_i = \sum_{j=1}^{J} a_{ij}$ is already a random walk of length J (described by the a_{ij}), on which we can apply the Local Central Limit Theorem and so $\Pr[a_i = 0] = \Theta(1/\sqrt{J})$. By noticing that summing independent random walks is equivalent to a longer one and applying the Local Central Limit Theorem, we have:

$$\Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0] \leq \Theta(1/\sqrt{J}) \ .$$

To explain further the two bounds:

- For the upper bound: we consider X fixed, and note $t = \sum_{i=1}^{\ell} a_i X_i$, by construction a_i are independent, so a_0 is independent from t then
$$\Pr[a(X') = 0] = \Pr[a_0 = -t] \leq \Pr[a_0 = 0] \leq \Theta(1/\sqrt{J})$$
using the above remark that a random walk is more likely to reach 0 than any other value, and a_0 is a random walk of length J.

- For the lower bound, we proceed by recurrence on ℓ, to show
$$H_\ell : \Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0] \quad (\text{where } X' \in 1 :: [\![0,\tau]\!]^\ell).$$
For $\ell = 0$, we consider $X' = 1$, we have a random walk of length J, so $\Theta(1/\tau\sqrt{J}) \leq \Theta(1/\sqrt{J}) \leq \Pr[a(X') = 0]$. We note $X_0 = 1$ for the hardwired 1 in X'. Let us suppose the property true at rank k, let us prove it at rank $k+1$:
  - If $\exists i_0, X_{i_0} = 0$ then we can consider a random walk of length k and apply the previous step, and conclude because $\Theta(1/\tau\sqrt{(k+1)J}) \leq \Theta(1/\tau\sqrt{kJ})$
  - Else, one can apply Lemma D.6.9 to conclude.

  Therefore, $\forall \ell, \forall X' \in 1 :: [\![0,\tau]\!]^\ell, \Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0]$.

We can now deduce that $\forall X, Y \in [\![0,\tau]\!]^\ell$ with $X \neq Y$: $\Pr[a(Y') = 0 | a(X') = 0] \leq \Theta(1/\sqrt{J})$. This can easily be seen by noting i_0 the first index where $Y_i \neq X_i$. We will note $\bar{X'} = X' - X_{i_0}$, in the following we will use the fact that $a(X') = 0 \Leftrightarrow a(\bar{X'}) = -a_{i_0} X_{i_0}$.[4]

$$\begin{aligned}
\Pr[a(Y') = 0 | a(X') = 0] &\leq \Pr[a(Y') = a(X') | a(X') = 0] \\
&\leq \Pr[Y_{i_0} a_{i_0} + a(\bar{Y'}) = X_{i_0} a_{i_0} + a(\bar{X'}) | a(X') = 0] \\
&\leq \max_t \Pr[(Y_{i_0} - X_{i_0}) a_{i_0} = t | a(\bar{X'}) = -X_{i_0} a_{i_0}] \qquad \text{(D.1)} \\
&\leq \max_{s,t'} \Pr[a_{i_0} = t' | a(\bar{X'}) = s] \qquad \text{(D.2)} \\
&\leq \max_{t'} \Pr[a_{i_0} = t'] \qquad \text{(D.3)} \\
&\leq \Pr[a_{i_0} = 0] \leq \Theta(1/\sqrt{J})
\end{aligned}$$

(D.1) We start with $(Y_{i_0} - X_{i_0}) a_{i_0} = a(\bar{X'}) - a(\bar{Y'})$, and then consider the maximum probability for all values $a(\bar{X'}) - a(\bar{Y'})$.

(D.2) We consider the maximum probability for all values of $-X_{i_0} a_{i_0}$.

(D.3) a_{i_0} and $a(\bar{X'})$ are independent.

Hence, for all $X_1, Y_1, \ldots, Y_q$, we have

$$\begin{aligned}
\Pr[a_{X_1} = 0 \wedge a_{Y_1}, \ldots, a_{Y_q} \neq 0] &= \Pr[a_{X_1} = 0] \Pr[a_{Y_1}, \ldots, a_{Y_q} \neq 0 | a_{X_1} = 0] \\
&\geq \Theta(1/\tau\sqrt{\ell J}) \left(1 - \sum_{i=1}^{q} \Pr[a_{Y_i} = 0 | a_{X_1} = 0]\right) \\
&\geq \Theta(1/\tau\sqrt{(\ell+1)J})(1 - q\Theta(1/\sqrt{J})) \ .
\end{aligned}$$

[4] $X \neq Y$ so i_0 exists, and thanks to the hardwired 1 we do not have to worry about Y' being a multiple of X'

Now we set $J = q^2$, to obtain the result. In that case the experiment success is lower-bounded by something linear in $1/(q\tau\sqrt{\ell+1})$. ▮

Studying the programmability of such functions is important. Having a $(q,1)$-programmable hash is a sufficient condition to instantiate a BLS-like signature scheme. Our result on the non-$(2,1)$ programmability over a non-binary alphabet while non-discarding the possibility says that it is probably not a good idea to try to instantiate such a scheme using Waters PHF. The $(1,q)$-programmability says that the Programmable Hash Function can be used in a signature scheme / IBE scheme, where we need to simulate q queries and use one challenge, this paper proposes a construction of such signature scheme and presents various applications.

D.A Asymmetric Version

All the schemes presented so far can be adapted for asymmetric groups. The main, and only difference, comes from the Groth-Sahai commitments. As symmetric bilinear groups are in general less efficient than *asymmetric* groups, we show how to instantiate our primitive with Groth-Sahai commitments in an asymmetric pairing-friendly group setting, relying on the SXDH assumption.

D.A.1 Assumptions

The security of Waters signatures in asymmetric bilinear groups was proven in [BFPV11] under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$.

Definition D.A.1 [Advanced Computational Diffie-Hellman problem (CDH^+)] Let $(\mathbb{G}_1, \mathbb{G}_2)$ be groups of prime order p with (g_1, g_2) as respective generators and e an admissible bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

ElGamal encryption is secure under the DDH assumption, which should hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$ for a more efficient variant of Groth-Sahai proofs to be secure.

Definition D.A.2 [Decisional Diffie-Hellman Assumption (DDH)] Let $\mathbb{G}$ be a cyclic group of prime order p. The DDH assumption states that given a 4-tuple $(g, g^a, g^b, g^c) \in \mathbb{G}$, it is hard to determine whether $c = ab$.

Definition D.A.3 [Symmetric external Diffie-Hellman Assumption (SXDH) [BBS04]] Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of the same prime order, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. The SXDH assumption states that DDH holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

D.A.2 Groth-Sahai Commitments

We will use SXDH-based Groth-Sahai commitments, which are a direct transposition of the previous ones in an asymmetric setting and replace double linear encryption by a double ElGamal encryption in a pairing friendly environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible bilinear map, for three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, generated by g_1, g_2 and $g_t = e(g_1, g_2)$ respectively.

The commitment key consists of four vectors $\mathbf{u}_1 = (u_{1,1}, u_{1,2}), \mathbf{u}_2 = (u_{2,1}, u_{2,2}) \in \mathbb{G}_1^2$ and $\mathbf{v}_1 = (v_{1,1}, v_{1,2}), \mathbf{v}_2 = (v_{2,1}, v_{2,2}) \in \mathbb{G}_2^2$. We write

$$\vec{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \quad \text{and} \quad \vec{\mathbf{v}} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix}.$$

- Binding initialization of the parameters is: $\vec{u_1} = (g_1, u)$ with $u = g_1^{\lambda}$ and $\vec{u_2} = \vec{u_1}^{\mu}$ with $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, which means that $\vec{\mathbf{u}}$ is a Diffie-Hellman tuple in $\mathbb{G}_1$, since $\vec{u_1} = (g_1, g_1^{\lambda})$ and $\vec{u_2} = (g_1^{\mu}, g_1^{\lambda\mu})$.
- Hiding initialization: we will use instead $\vec{u_2} = \vec{u_1}^{\mu} \odot (1, g_1)^{-1}$: $\vec{u_1} = (g_1, g_1^{\lambda})$ and $\vec{u_2} = (g_1^{\mu}, g_1^{\lambda\mu-1})$, and analogously for in $\mathbb{G}_2$ for $\vec{v}$.

Under the SXDH assumption, the two initializations are indistinguishable.

Commitments to Group Elements. To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets

$$\mathcal{C}(X) = (1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} = (u_{1,1}^{s_1} \cdot u_{2,1}^{s_2}, X \cdot u_{1,2}^{s_1} \cdot u_{2,2}^{s_2}) \ .$$

A simulator that knows the discrete logarithm λ of u in basis g_1 can extract X in the perfectly binding setting. The commitment in $\mathbb{G}_2$ follows the same rules, with $\vec{v}$ and g_2 instead of $\vec{u}$ and g_1.

Commitments to Scalars. One actually commits to g_1^x, from which x can be extracted if this is a bit.

Proofs. This time, a Groth-Sahai proof is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. One has to pay attention to the fact that Groth-Sahai bit-by-bit proofs in SXDH require bits to be committed both in $\mathbb{G}_1$ and $\mathbb{G}_2$ and thus require to use 2 quadratic equations by bit.

D.A.3 Partially Blind Signatures with Perfect Blindness

The construction is completely straightforward. If we follow the steps from the DLin-version: We will need 2 group elements for the commitment to M in $\mathbb{G}_1$, 4 group elements to commit Y_1, Y_2 in $\mathbb{G}_1$, the proofs will require 4 group elements in $\mathbb{G}_2$. We will need 6ℓ elements in each group to commit to M and prove we indeed committed it bit-by-bit, and 2 extra group elements in $\mathbb{G}_2$ to prove c_2 is well-formed. The signatures on the committed elements will require 3 groups elements in $\mathbb{G}_1$ and one in $\mathbb{G}_2$. Therefore the overall scheme will require $(6\ell + 9, 6\ell + 7)$ group elements communication.

Appendix E

Fair Blind Signatures without Random Oracles

Africacrypt 2010
[FV10] with G. Fuchsbauer

Abstract : A fair blind signature is a blind signature with revocable anonymity and unlinkability, i.e. an authority can link an issuing session to the resulting signature and trace a signature to the user who requested it. In this paper we first revisit the security model for fair blind signatures given by Hufschmitt and Traoré in 2007. We then give the first practical fair blind signature scheme with a security proof in the standard model. Our scheme satisfies a stronger variant of the Hufschmitt-Traoré model.

E.1 Introduction

A blind signature scheme is a protocol for obtaining a signature from an issuer (signer) such that the issuer's view of the protocol cannot be linked to the resulting message/signature pair. Blind signatures are employed in privacy-related protocols where the issuer and the message author are different parties (e.g., e-voting or e-cash systems). However, blind signature schemes provide perfect unlinkability and could therefore be misused by dishonest users. Fair blind signatures were introduced by Stadler, Piveteau and Camenisch [SPC95] to prevent abuse of unlinkability. They allow two types of blindness revocation: linking a signature to the user who asked for it and identifying a signature that resulted from a given signing session. A security model for fair blind signatures was introduced by Hufschmitt and Traoré [HT07].

We first revisit this security model and propose a stronger variant. We then present the first efficient fair blind signature scheme with a standard-model security proof (i.e. without resorting to the random-oracle heuristic) in the strengthened model. We make extensive use of the non-interactive proof system due to Groth and Sahai [GS08] and of the *automorphic signatures* recently introduced by Fuchsbauer [Fuc09]; we do not rely on interactive assumptions.

E.1.1 Prior work

The concept of *blind signatures* was introduced by Chaum in [Cha82]. A blind signature scheme is a cryptographic primitive that allows a user to obtain from the *issuer* (signer) a digital signature on a message of the user's choice in such a way that the issuer's view of the protocol

cannot be linked to the resulting message/signature pair. Blind signatures have numerous applications including e-cash: they prevent linking withdrawals and payments made by the same customer. However, the impossibility of this linking might lead to frauds (money laundering, blackmailing, ...); some applications therefore require means to identify the resulting signature from the transcript of a signature-issuing protocol or to link a message/signature pair to user who requested it.

Fair blind signatures were introduced by Stadler, Piveteau and Camenisch in [SPC95] to provide these means. Several fair blind signature schemes have been proposed since then [SPC95, AO01, HT07] with applications to e-cash [GT03] or e-voting [CGT06]. In [HT07], Hufschmitt and Traoré presented the first formal security model for fair blind signatures and a scheme based on bilinear maps satisfying it in the random oracle model under an interactive assumption. In a recent independent work, Rückert and Schröder [RS10] proposed a *generic* construction of fair *partially* blind signatures [AF96].

E.1.2 Our contribution

As a first contribution, we strengthen the security model proposed in [HT07]. In our model, opening a transcript of an issuing session not only reveals information to identify the resulting signature, but also the user that requested it.

We give a definition of blindness analogously to [Oka06], but additionally provide tracing oracles to the adversary; in contrast to [HT07], this models *active* adversaries. We propose a traceability notion that implies the original one. Finally, we formalize the non-frameability notions analogously to [BSZ05], where it is the adversary's task to output a framing signature (or transcript) *and a proof*. (In [HT07] the experiment produces the proof, limiting thus the adversary.) We believe that our version of signature non-frameability is more intuitive: no corrupt issuer can output a transcript, an opening framing a user, and a proof. (In [HT07] the adversary must output a message/signature pair such that an honest transcript opens to it.) (See §E.2.3 for the details.)

In 2008, Groth and Sahai [GS08] proposed a way to produce efficient non-interactive zero-knowledge (NIZK) and non-interactive witness-indistinguishable (NIWI) proofs for (algebraic) statements related to groups equipped with a bilinear map. In particular, they give proofs of satisfiability of *pairing-product equations* (cf. §E.4.2 and [BFI+10] for efficiency improvements for proof verification). In [Fuc09], Fuchsbauer introduced the notion of *automorphic signatures* whose verification keys lie in the message space, messages and signatures consist of group elements only, and verification is done by evaluating a set of pairing-product equations (cf. §E.5). Among several applications, he constructed an (automorphic) blind signature in the following way: the user commits to the message, and gives the issuer a randomized message; the issuer produces a "pre-signature" from which the user takes away the randomness to recover a signature. The actual signature is then a Groth-Sahai NIWI proof of knowledge of a signature, which guarantees unlinkability to the issuing.

In this paper, we modify Fuchsbauer's blind signature scheme in order to construct the first practical fair blind signature scheme with a security reduction in the standard model. Our security analysis does not introduce any new computational assumptions and relies only on falsifiable assumptions [Nao03] (cf. §E.3). First, we extend Fuchsbauer's automorphic signature so it can sign three messages at once. Then, to achieve blindness even against adversaries provided with tracing oracles, we use Groth's technique from [Gro07] to achieve CCA-anonymous group signatures: instead of just committing to the tracing information, we additionally encrypt it (using Kiltz' tag-based encryption scheme [Kil06]) and provide NIZK proofs of consistency with the commitments. In order to achieve the strengthened notion of non-frameability, we construct simulation-sound NIZK proofs of knowledge of a Diffie-Hellman solution which consist

of group elements only and are verified by checking a set of pairing-product equations (i.e. they are Groth-Sahai compatible).

Since messages and signatures consist of group elements only and their verification predicate is a conjunction of pairing-product equations, our fair blind signatures are Groth-Sahai compatible themselves which makes them perfectly suitable to design efficient fair e-cash systems following the approach proposed in [GT03]. In addition, our scheme is compatible with the "generic" variant[1] of Votopia [OMA+99] proposed by Canard, Gaud and Traoré in [CGT06]. Combined with a suitable mix-net (e.g. [GL07]), it provides a practical electronic voting protocol in the standard model including public verifiability, and compares favorably with other similar systems in terms of computational cost.

E.2 The Model

E.2.1 Syntax

Definition E.2.1 A *fair blind signature scheme* is a 10-tuple

$$(\mathsf{Setup}, \mathsf{IKGen}, \mathsf{UKGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Ver}, \mathsf{TrSig}, \mathsf{TrId}, \mathsf{ChkSig}, \mathsf{ChkId})$$

of (interactive) (probabilistic) polynomial-time Turing machines ((P)PTs):

Setup is a PPT that takes as input an integer λ and outputs the *parameters* pp and the *revocation key* rk. We call λ the *security parameter*.

IKGen is a PPT that takes as input the parameters pp and outputs a pair (ipk, isk), the *issuer's public and secret key*.

UKGen is a PPT that takes as input the parameters pp and outputs a pair (upk, usk), the *user's public and secret key*.

Sign and User are interactive PPTs such that User takes as inputs pp, the issuer's public key ipk, the user's secret key usk and a bit string m; Sign takes as input pp, the issuer's secret key isk and user public key upk. Sign and User engage in the *signature-issuing protocol* and when they stop, Sign outputs `completed` or `not-completed` while User outputs $\perp$ or a bit string σ.

Ver is a deterministic PT (DPT) that on input the parameters pp, an issuer public key ipk and a pair of bit strings (m, σ) outputs either 0 or 1. If it outputs 1 then σ is a *valid signature* on the *message* m

TrSig is a DPT that on input pp, an issuer public key ipk, a transcript *trans* of a signature-issuing protocol and a revocation key rk outputs three bit strings (upk, id_σ, π).

TrId is a DPT that on input pp, an issuer public key ipk, a message/signature pair (m, σ) for ipk and a revocation key rk outputs two bit strings (upk, π).

ChkSig is a DPT that on input pp, an issuer public key ipk, a transcript of a signature issuing protocol, a pair message/signature (m, σ) for ipk and three bit strings (upk, id_σ, π), outputs either 0 or 1.

ChkId is a DPT that on input pp, an issuer public key ipk, a message/signature pair (m, σ) for ipk and two bit strings (upk, π), outputs either 0 or 1.

[1] This variant was used during the French referendum on the European Constitution in May 2005.

For all $\lambda \in \mathbb{N}$, all pairs (pp, rk) output by $\mathsf{Setup}(\lambda)$ all pairs (ipk, isk) output by $\mathsf{IKGen}(pp)$, and all pairs (upk, usk) output by $\mathsf{UKGen}(pp)$:

1. if Sign and User follow the signature-issuing protocol with respective inputs (pp, isk, upk) and (pp, usk, ipk, m), then Sign outputs `completed` and User outputs a bit string σ that satisfies $\mathsf{Ver}(ipk, (m, \sigma)) = 1$;

2. on input ipk, the transcript $trans$ of the protocol and rk, TrSig outputs three bit strings (upk, id_σ, π) s.t. $\mathsf{ChkSig}(pp, ipk, trans, (m, \sigma), (upk, id_\sigma, \pi)) = 1$;

3. on input ipk, the pair (m, σ) and rk, TrId outputs two bit strings (upk, π) such that $\mathsf{ChkId}(pp, ipk, (m, \sigma), (upk, \pi)) = 1$.

E.2.2 Security Definitions

To define the security notions for fair blind signatures, we use a notation similar to the one in [BSZ05] used in [HT07]:

HU denotes the set of honest users and CU is the set of corrupted users.

AddU is an *add-user* oracle. The oracle runs $(upk, usk) \leftarrow \mathsf{UKGen}(pp)$, adds upk to HU and returns it to the adversary.

CrptU is a *corrupt-user* oracle. The adversary calls it with a pair (upk, usk) and upk is added to the set CU.

USK is a *user-secret-key* oracle enabling the adversary to obtain the private key usk for some $upk \in HU$. The oracle transfers upk to CU and returns usk.

User is an *honest-user* oracle. The adversary impersonating a corrupt issuer calls it with (upk, m). If $upk \in HU$, the experiment simulates the honest user holding upk running the signature issuing protocol with the adversary for message m. If the issuing protocol completed successfully, the adversary is given the resulting signature. The experiment keeps a list Set with entries of the form $(upk, m, trans, \sigma)$, to record an execution of User, where $trans$ is the transcript of the issuing session and σ is the resulting signature. (Note that only valid σ's (i.e. the protocol was successful) are written to Set.

Sign is a *signing* oracle. The adversary impersonating a corrupt user can use it to run the issuing protocol with the honest issuer. The experiment keeps a list $Trans$ in which the transcripts $trans_i$ resulting from Sign calls are stored.

$\mathsf{Challenge}_b$ is a *challenge* oracle, which (w.l.o.g.) can only be called once. The adversary provides two user public keys upk_0 and upk_1 and two messages m_0 and m_1. The oracle first simulates User on inputs (pp, ipk, usk_b, m_b) and then, in a second protocol run, simulates User on inputs $(pp, ipk, usk_{1-b}, m_{1-b})$. Finally, the oracle returns (σ_0, σ_1), the resulting signatures on m_0 and m_1.

TrSig (resp. TrId) is a *signature (resp. identity) tracing* oracle. When queried on the transcripts (or messages) emanating from a $\mathsf{Challenge}$ call, they return $\perp$.

Figure E.1 formalizes the experiments for the following security notions:

Blindness. Not even the issuer with access to tracing oracles can link a message/signature pair to the signature-issuing session it stems from.

Identity Traceability. No coalition of users can produce a set of signatures containing signatures which cannot be linked to an identity.

Signature Traceability. No coalition of users is be able to produce a message/signature pair which is not traced by any issuing transcript or two pairs which are traced by the same transcript.

Identity Non-Frameability. No coalition of issuer, users and tracing authority should be able to provide a signature and a proof that the signature opens to an honest user who did not ask for the signature.

Signature Non-Frameability. No coalition of issuer, users and tracing authority should be able to provide a transcript that either wrongfully opens to an honest signature or an honest user.

We say that a fair blind signature achieves *blindness* if for all p.p.t. adversaries $\mathcal{A}$, the following is negligible: $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{blind-1}} = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{blind-0}} = 1] - \frac{1}{2}$. The remaining security notions are achieved if for all p.p.t. $\mathcal{A}$, the probability that the corresponding experiment returns 1 is negligible.

E.2.3 A Note on the Hufschmitt-Traoré Security Notions

Blindness.

In [HT07], the challenge oracle (called "Choose") is defined as follows: the adversary provides two user public keys upk_0 and upk_1 and a message, and obtains a signature under upk_b. This gives a weak security guarantee, as the adversary—who should impersonate the issuer—cannot actively participate in the issuing of the challenge signature. We define our oracle in the spirit of [Oka06]: the adversary chooses two users (and messages) which interact with him in random order; he gets to see both resulting signatures and has to determine the order of issuing.

Traceability Notions.
Intuitively, identity traceability means that no coalition of users and the authority can create a message/signature pair that is not traceable to a user, which is what was formalized in [HT07].

We propose the following experiment leading to a stronger notion: the adversary gets the authority's key and impersonates corrupt users, who, via the Sign oracle can request signatures from the honest issuer. The latter is simulated by the experiment and keeps a set *Trans* of transcripts of oracle calls. Eventually, the adversary outputs a set of message/signature pairs. The experiment opens all transcripts to get a list of users to which signatures were issued. Another list of users is constructed by opening the returned signatures. The adversary wins if there exists a user who appears more often in the second list than in the first, or if $\perp$ is in the second list, or if any of the proofs output by the opening algorithm do not verify. Note that the notion of [HT07] is implied by ours.

Non-Frameability Notions.
Non-frameability means that not even a coalition of everyone else can "frame" an honest user. For example, no adversary can output a signature which opens to a user who did not participate in its issuing. In [HT07], the adversary outputs a message/signature pair, which is then opened by the experiment to determine if it "framed" a user. Analogously to [BSZ05] (who defined non-frameability for group signatures), we define a strictly stronger notion requiring the adversary to output an incriminating signature, an honest user, *and a valid proof* that the signature opens

to that user. Note that only this formalization makes the π output by the tracing algorithms a proof, as it guarantees that no adversary can produce a proof that verifies for a false opening.

Identity Non-Frameability. In [HT07], the adversary wins if it produces a pair (m, σ) such that, when opened to *upk*, we have $(m, \sigma, upk) \notin Set$. This seems to guarantee *strong* unforgeability where an adversary modifying a signature returned by the experiment wins the game. This is however not the case in the scheme proposed in [HT07]: the final signature is a proof of knowledge of some values computed by the issuer made non-interactive by the Fiat-Shamir heuristic; hence from a given signature issuing session the user may derive several valid signatures on a message m. For that reason, the model in [HT07] considers two signatures different only if the underlying secrets are different. We adopt the same convention in this paper in that we consider two signatures equivalent if they have the same (public) *identifier*.

Signature Non-Frameability. Non-frameability of signature tracing intuitively means: even if everyone else colludes against an honest user, they cannot produce a transcript that opens to an honest signature. In the definition proposed in [HT07], the adversary plays the issuer in that he gets his secret key. However, he has no possibility to communicate with honest users since the *challenger* plays the issuer in the signature-issuing sessions with honest users and the adversary only gets the transcripts. His goal is to produce a *new* message/signature pair (one that does not emanate from a User-oracle call) such that an honest transcript opens to it.

We give the following security notion which we think is more intuitive. No corrupt issuer can produce a transcript of an issuing session and one of the following: either a public key of an honest user and a proof that this user participated in the transcript whereas she did not; or a signature identifier of an honest signature coming from a different session and a proof that the transcript opens to it. Similarly to signatures we consider two transcripts equivalent if the contain the same user randomness and the same issuer randomness.

Unforgeability.
Consider an adversary that breaks the classical security notion for blind signatures, one-more unforgeability, i.e. after $q-1$ Sign-oracle queries, he outputs q signatures on different messages. We show that the adversary must have broken signature traceability: indeed since there are more signatures than transcripts, either there is a signature which no transcripts points to, or there is a transcript that points to two signatures.

E.3 Assumptions

A (symmetric) *bilinear group* is a tuple $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ where $(\mathbb{G}, \cdot)$ and $(\mathbb{G}_T, \cdot)$ are two cyclic groups of prime order p, G is a generator of $\mathbb{G}$, and $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate bilinear map, i.e. $\forall U, V \in \mathbb{G}\ \forall a, b \in \mathbb{Z}\colon\ e(U^a, V^b) = e(U, V)^{ab}$, and $e(G, G)$ is a generator of $\mathbb{G}_T$.

The *Decision Linear (DLIN) Assumption* [BBS04], in $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ states that given a 5-tuple $(G^\alpha, G^\beta, G^{r\alpha}, G^{s\beta}, G^t)$ for random $\alpha, \beta, r, s \in \mathbb{Z}_p$, it is hard to decide whether $t = r + s$ or t is random.

The following two assumptions were introduced by [FPV09] and [Fuc09], respectively. Under the *knowledge of exponent assumption* [Dam91], the first is equivalent to SDH [BB04b] and the second is equivalent to computing discrete logarithms.

Assumption E.3.1 [q-DHSDH] Given $(G, H, K, X = G^x) \in \mathbb{G}^4$ and $q-1$ tuples

$$\left(A_i = (KG^{v_i})^{\frac{1}{x+d_i}},\ C_i = G^{d_i},\ D_i = H^{d_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right)_{i=1}^{q-1},$$

for $d_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to output a new tuple $(A, C, D, V, W) \in \mathbb{G}^5$ satisfying

$$e(A, XC) = e(KV, G) \qquad e(C, H) = e(G, D) \qquad e(V, H) = e(G, W) \tag{E.1}$$

The next assumption states that, given $(G, H, T) \in \mathbb{G}^3$, it is hard to produce a non-trivial (G^m, H^m, G^r, H^r) such that $G^m = T^r$.

Assumption E.3.2 [HDL] Given a random triple $(G, H, T) \in \mathbb{G}^3$, it is hard to output a 4-tuple $(M, N, R, S) \neq (1, 1, 1, 1)$ such that

$$e(R, T) = e(M, G) \qquad e(M, H) = e(G, N) \qquad e(R, H) = e(G, S) \tag{E.2}$$

E.4 Tools

We recall some tools from the literature which we use to construct our scheme.

E.4.1 A Signature Scheme to Sign Group Elements

We present the signature scheme from [Fuc09], which is secure against chosen-message attacks under Assumptions E.3.1 and E.3.2. Its message space is the set of *Diffie-Hellman pairs* $\mathcal{DH} := \{(A, B) \in \mathbb{G}^2 \,|\, \exists \alpha : A = G^\alpha, B = H^\alpha\}$ w.r.t. two fixed generators $G, H \in \mathbb{G}$. Note that $(A, B) \in \mathcal{DH}$ iff $e(A, H) = e(G, B)$.

Scheme E.4.1 [$\mathbf{Sig}_1$]

Setup_1 Given $(p, \mathbb{G}, \mathbb{G}_T, e, G)$, choose additional generators $H, K, T \in \mathbb{G}$.

KeyGen_1 Choose $sk = x \leftarrow \mathbb{Z}_p$ and set $vk = G^x$.

Sign_1 To sign $(M, N) \in \mathcal{DH}$ with secret key x, choose $d, r \leftarrow \mathbb{Z}_p$ and output

$$\left(A := (KT^r M)^{\frac{1}{x+d}}, C := G^d, D := H^d, R := G^r, S := H^r\right) ,$$

Verify_1 (A, C, D, R, S) is valid on $(M, N) \in \mathcal{DH}$ under public key $vk = X$ iff

$$e(A, XC) = e(KM, G)\, e(T, R) \qquad \begin{matrix} e(C, H) = e(G, D) \\ e(R, H) = e(G, S) \end{matrix} \tag{E.3}$$

E.4.2 Groth-Sahai Proofs

We sketch the results of Groth and Sahai [GS08] on proofs of satisfiability of sets of equations over a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, G)$. Due to the complexity of their methodology, we present what is needed for our results and refer to the full version of [GS08] for any additional details.

We define a key for *linear commitments*. Choose $\alpha, \beta, r_1, r_2 \leftarrow \mathbb{Z}_p$ and define $U = G^\alpha$, $V = G^\beta$, $W_1 := U^{r_1}$, $W_2 := V^{r_2}$, and W_3 which is either

- soundness setting: $W_3 := G^{r_1 + r_2}$ (which makes $\vec{\mathbf{u}}$ a binding key); or
- witness-indistinguishable setting: $W_3 := G^{r_1 + r_2 - 1}$ (making $\vec{\mathbf{u}}$ a hiding key)

Under key $ck = (U, V, W_1, W_2, W_3)$, a commitment to a group element $X \in \mathbb{G}$ using randomness $(s_1, s_2, s_3) \leftarrow \mathbb{Z}_p^3$ is defined as

$$\mathsf{Com}(ck, X; (s_1, s_2, s_3)) := (U^{s_1} W_1^{s_3},\ V^{s_2} W_2^{s_3},\ X G^{s_1 + s_2} W_3^{s_3}) .$$

In the soundness setting, given the *extraction key* $ek := (\alpha, \beta)$, the committed value can be extracted from a commitment $\mathbf{c} = (c_1, c_2, c_3)$. On the other hand, in the witness-indistinguishable

(WI) setting, $\mathbf{c}$ is equally distributed for every X. The two settings are indistinguishable under the DLIN assumption.

A *pairing-product equation* is an equation for variables $\mathcal{Y}_1, \ldots, \mathcal{Y}_n \in \mathbb{G}$ of the form

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}} \;=\; t_T \;,$$

with $\mathcal{A}_i \in \mathbb{G}$, $\gamma_{i,j} \in \mathbb{Z}_p$ and $t_T \in \mathbb{G}_T$ for $1 \leq i, j \leq n$.

To prove satisfiability of a set of equations of this form, one first makes commitments to a satisfying witness (i.e. an assignment to the variables of each equation) and then adds a "proof" per equation. Groth and Sahai describe how to construct these: they are in $\mathbb{G}^{3\times 3}$ (or $\mathbb{G}^3$ when all $\gamma_{i,j} = 0$). In the soundness setting, if the proof is valid then Extr extracts the witness satisfying the equations. In the WI setting, commitments and proofs of different witnesses which both satisfy the same pairing-product equation are equally distributed.

E.4.3 Commit and Encrypt

In order to build CCA-anonymous group signatures, Groth [Gro07] uses the following technique: a group signature consists of linear commitments to a certified signature and Groth-Sahai proofs that the committed values constitute a valid signature. CPA-anonymity follows from WI of GS proofs: once the commitment key has been replaced by a perfectly hiding one, a group signature reveals no information about the signer. However, in order to simulate opening queries in the WI setting, some commitments are doubled with a *tag-based encryption* under Kiltz' scheme [Kil06] and a Groth-Sahai NIZK proof that the committed and the encrypted value are the same. To produce a group signature, the user first chooses a key pair for a one-time signature scheme, uses the verification key as the tag for the encryption and the secret key to sign the group signature.

By $\mathbf{Sig}_{\mathrm{ot}} = (\mathsf{KeyGen}_{\mathrm{ot}}, \mathsf{Sign}_{\mathrm{ot}}, \mathsf{Ver}_{\mathrm{ot}})$ we will denote the signature scheme discussed in §E.5.2 which satisfies the required security notion. By CEP (commit-encrypt-prove) we denote the following:

$$\mathsf{CEP}(ck, pk, tag, msg;\, (\rho, r)) := \\ (\mathsf{Com}(ck, msg;\, \rho),\, \mathsf{Enc}(pk, tag, msg;\, r),\, \mathsf{NizkEq}(ck, pk,\, tag;\, msg, \rho, r))$$

where Enc denotes Kiltz' encryption and NizkEq denotes a Groth-Sahai NIZK proof that the commitment and the encryption contain the same plaintext (cf. [Gro07]). We say that an output $\psi = (\mathbf{c}, C, \zeta)$ of CEP is *valid* if the ciphertext and the zero-knowledge proof are valid.

E.5 New Tools

E.5.1 A Scheme To Sign Three Diffie-Hellman Pairs

We extend the scheme from §E.4.1, so it signs three messages at once; we prove existential unforgeability (EUF) against adversaries making a particular chosen message attack (CMA): the first message is given (as usual) as a Diffie-Hellman pair, whereas the second and third message are queried as their logarithms; that is, instead of querying (G^v, H^v), the adversary has to give v explicitly. As we will see, this combines smoothly with our application.

Scheme E.5.1 [$\mathbf{Sig}_3$]

$\mathsf{Setup}_3(\mathcal{G})$ Given $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, e, G)$, choose additional generators $H, K, T \in \mathbb{G}$.

$\mathsf{KeyGen}_3(\mathcal{G})$ Choose $sk = (x, \ell, u) \leftarrow \mathbb{Z}_p^3$ and set $vk = (G^x, G^\ell, G^u)$.

$\mathsf{Sign}_3((x, \ell, u), (M, N, Y, Z, V, W))$ A signature on $((M, N), (Y, Z), (V, W)) \in \mathcal{DH}^3$ under public key G^x, is defined as (for random $d, r \leftarrow \mathbb{Z}_p$)

$$\left(A := (KT^r MY^\ell V^u)^{\frac{1}{x+d}}, C := G^d, D := H^d, R := G^r, S := H^r\right)$$

Verify_3 (A, C, D, R, S) is valid on messages $(M, N), (Y, Z), (V, W) \in \mathcal{DH}$ under a public key (X, L, U) iff

$$e(A, XC) = e(KM, G)\, e(T, R)\, e(L, Y)\, e(U, V) \qquad \begin{array}{l} e(C, H) = e(G, D) \\ e(R, H) = e(G, S) \end{array} \tag{E.4}$$

Theorem E.5.2 $\mathbf{Sig}_3$ is existentially unforgeable against adversaries making chosen message attacks of the form $((M_1, N_1), m_2, m_3)$.

Proof: Let (M_i, N_i, y_i, v_i) be the queries, $(A_i, C_i, D_i, R_i = G^{r_i}, S_i)$ be the responses. Let (M, N, Y, Z, V, W) and $(A, C, D, R = G^r, S)$ be a successful forgery. We distinguish 4 types of forgers (where $Y_i := G^{y_i}, V_i := G^{v_i}$):

Type I
$$\forall\, i : T^{r_i} M_i Y_i^\ell V_i^u \neq T^r MY^\ell V^u \tag{E.5}$$

Type II
$$\exists\, i : T^{r_i} M_i Y_i^\ell V_i^u = T^r MY^\ell V^u \ \wedge\ M_i Y_i^\ell V_i^u \neq MY^\ell V^u \tag{E.6}$$

Type III
$$\exists\, i : M_i Y_i^\ell V_i^u = MY^\ell V^u \ \wedge\ M_i V_i^u \neq MV^u \tag{E.7}$$

Type IV
$$\exists\, i : M_i Y_i^\ell V_i^u = MY^\ell V^u \ \wedge\ M_i V_i^u = MV^u \tag{E.8}$$

Type I is reduced to DHSDH. Let $(G, H, K, (A_i, C_i, D_i, E_i, F_i)_{i=1}^{q-1})$ be an instance. Choose and $t, \ell, u \leftarrow \mathbb{Z}_p$ and set $T = G^t$, $L = G^\ell$ and $U = G^u$. A signature on an 8-tuple $(M_i, N_i, Y_i, Z_i, y_i, V_i, W_i, v_i)$ is (after a consistency check) answered as

$$(A_i, C_i, D_i, (E_i M_i^{-1} Y_i^{-\ell} V_i^{-u})^{1/t}, (F_i N_i^{-1} Z_i^{-\ell} W_i^{-u})^{1/t}).$$

After a successful forgery, return $(A, C, D, R^t MY^\ell V^u, S^t NZ^\ell W^u)$, which is a valid DHSDH solution by (E.5).

Type II is reduced to HDL. Let (G, H, T) be an HDL instance. Generate the rest of the parameters and a public key and answer the queries by signing. After a successful forgery return the following, which is non-trivial by (E.6):

$$(MY^\ell V^u M_i^{-1} Y_i^{-\ell} V_i^{-u}, NZ^\ell W^u N_i^{-1} Z_i^{-\ell} W_i^{-u}, R_i R^{-1}, S_i S^{-1})\ .$$

Type III is reduced to HDL. Let (G, H, L) be an instance. Choose $K, T \leftarrow \mathbb{G}$ and $x, u \leftarrow \mathbb{Z}_p$ and return the parameters and public key $(X = G^x, L, U = G^u)$. Thanks to the y_i in the signing queries, we can simulate them: return $((KT^{r_i} M_i L^{y_i} V_i^u)^{\frac{1}{x+d_i}}, G^{d_i}, H^{d_i}, G^{r_i}, H^{r_i})$. We have $MV^u M_i^{-1} V_i^{-u} = Y_i^\ell Y^{-\ell} = L^{y_i - y}$ from (E.7), so from a successful forgery, we can return

$$(MV^u M_i^{-1} V_i^{-u}, NW^u N_i^{-1} W_i^{-u}, Y_i Y^{-1}, Z_i Z^{-1})\ ,$$

which is non-trivial by (E.7).

Type IV is also reduced to HDL. Let (G, H, U) be an HDL instance. Choose $K, T \leftarrow \mathbb{G}$ and $x, \ell \leftarrow \mathbb{Z}_p$ and return the parameters and public key $(X = G^x, L = G^\ell, U)$. Thanks to the v_i in the signing queries, we can simulate the signatures by returning the tuple $((KT^{r_i} M_i Y_i^\ell U^{v_i})^{\frac{1}{x+d_i}}, G^{d_i}, H^{d_i}, G^{r_i}, H^{r_i})$. From a successful forgery of Type IV we have $MM_i^{-1} = U^{v_i - v}$ from (E.7), we can thus return $(MM_i^{-1}, NN_i^{-1}, V_i V^{-1}, W_i W^{-1})$, which is non-trivial, (M, N, Y, Z, V, W) being a valid forgery and $(Y, Z) = (Y_i, Z_i)$ by (E.8).

∎

E.5.2 A Simulation-Sound Non-Interactive Zero-Knowledge Proof of Knowledge of a CDH Solution

Let (G, F, V) be elements of $\mathbb{G}$. We construct a simulation-sound non-interactive zero-knowledge (SSNIZK) proof of knowledge (PoK) of W s.t. $e(V, F) = e(G, W)$. We follow the overall approach by Groth [Gro06]. The common reference string (CRS) contains a CRS for Groth-Sahai (GS) proofs and a public key for a EUF-CMA signature scheme **Sig**. A proof is done as follows: choose a key pair for a one-time signature scheme $\mathbf{Sig}_{\text{ot}}$, and make a witness-indistinguishable GS proof of the following: either to know W, a CDH solution for (G, F, V) or to know a signature on the chosen one-time key which is valid under the public key from the CRS;[2] finally sign the proof using the one-time key. A SSNIZKPoK is verified by checking the GS proofs and the one-time signature. Knowing the signing key corresponding to the key in the CRS, one can simulate proofs by using as a witness a signature on the one-time key.

We require that a proof consist of group elements only and is verified by checking a set of pairing-product equations. This can be achieved by using Scheme E.4.1 and a one-time scheme to sign group elements using the commitment scheme in [Gro09] based on the DLIN assumption.[3]

E.6 A Fair Blind Signature Scheme

The basis of our protocol is the blind automorphic signature scheme from [Fuc09]: the user randomizes the message to be signed, the issuer produces a pre-signature from which the user obtains a signature by removing the randomness; the final signature is a Groth-Sahai (GS) proof of knowledge of the resulting signature.

In our scheme, in addition to the message, the issuer signs the user's public key, and an *identifier* of the signature, which the issuer and the user define jointly. Note that the issuer may neither learn the user's public key nor the identifier. To guarantee provable tracings, the user signs what she sends in the issuing protocol and the final signature. To prevent malicious issuers from producing a transcript that opens to an honest signature, the proof contains a SSNIZK proof of knowledge of the randomness introduced by the user. To achieve blindness against adversaries with tracing oracles, the elements that serve as proofs of correct tracing are additionally encrypted and the transcript (and final signature) is signed with a one-time key (cf. §E.4.3).

[2] [Gro06] shows how to express a disjunction of equation sets by a new set of equations.

[3] The strong one-time signature scheme from [Gro06] works as follows: the verification key is an (equivocable) Pedersen commitment to 0; to sign a message, the commitment is opened to the message using the trapdoor; putting a second trapdoor in the commitment scheme, we can simulate one signing query and use a forger to break the binding property of the commitment. In [Gro09], Groth proposes a scheme to commit to group elements which is computationally binding under DLIN. Using his scheme instead of Pedersen commitments, we can construct an efficient one-time signature on group elements s.t. signatures consist of group elements. Using his scheme rather than Pedersen commitments, we can construct an efficient one-time signature scheme for group elements whose signatures consist of group elements (see Appendix E.A).

To open a signature (i.e. to trace a user), the authority extracts tracing information from the commitments as well as signatures that act as proofs.

E.6.1 A Blind Signature Scheme

Setup. Choose a bilinear group $\mathcal{G} := (p, \mathbb{G}, \mathbb{G}_T, e, G)$ and parameters (H, K, T) for $\mathbf{Sig}_3$. Pick $F, H' \leftarrow \mathbb{G}$, a commitment and extraction key (ck, ek) for GS proofs, a key pair for tag-based encryption (epk, esk) and $sscrs$, a common reference string for $\mathsf{SSNIZKPoK}$. Output $pp := (\mathcal{G}, G, H, K, T, F, H', ck, epk, sscrs)$ and $rk := ek$.

Key Generation. Both IKGen and UKGen are defined as KeyGen, i.e. the key generation algorithm for $\mathbf{Sig}_1$.

Signature Issuing. The common inputs are $(pp, ipk = G^x)$, the issuer's additional input is $isk = x$, the user's inputs are $(upk = G^y, usk = y,\ (M, N) \in \mathcal{DH})$.

User Choose $\eta, v' \leftarrow \mathbb{Z}_p$ and set $P = G^\eta, Q = F^\eta, V' = G^{v'}, W' = F^{v'}$.
Produce $\xi \leftarrow \mathsf{SSNIZKPoK}(sscrs, (P, V'), (Q, W'))$. [4]
Choose $(vk'_{\text{ot}}, sk'_{\text{ot}}) \leftarrow \mathsf{KeyGen}_{\text{ot}}(\mathcal{G})$ and set $\Sigma' \leftarrow \mathsf{Sign}(usk, vk'_{\text{ot}})$. [5]
Send the following

1. $Y = G^y, Z = H^y$, vk'_{ot}, Σ';
2. $\mathbf{c}_M = \mathsf{Com}(ck, M)$; $\mathbf{c}_N := \mathsf{Com}(ck, N)$,
   $\psi_P, \psi_V, \vec{\psi}_\xi$, with $\psi_\odot := \mathsf{CEP}(ck, epk, vk'_{\text{ot}}, \odot)$,
   a proof ϕ_M that $(M, N) \in \mathcal{DH}$ and a proof ϕ_ξ of validity of ξ;
3. $J := (KML^yU^{v'})^{\frac{1}{\eta}}$;
4. a zero-knowledge proof ζ of knowledge of η, y and v' such that
   - $Y = G^y$,
   - $\mathbf{c}_V$ commits to $G^{v'}$, and
   - $\mathbf{c}_M$ commits to $J^\eta L^{-y} U^{-v'} K^{-1}$;
5. $sig' \leftarrow \mathsf{Sign}_{\text{ot}}(sk'_{\text{ot}}, (Y, Z, \Sigma', \mathbf{c}_M, \mathbf{c}_N, \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, J, \zeta, vk'_{\text{ot}}))$.

Issuer If $\Sigma', \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, sig'$ and the proof of knowledge are valid, choose $d, r, v'' \leftarrow \mathbb{Z}_p$ and send:

$$A' := (JT^rU^{v''})^{\frac{1}{x+d}} \qquad C := G^d \qquad D := F^d \qquad R' := G^r \qquad S' := H^r \qquad v''$$

The user does the following:

1. set $A := (A')^\eta, R := (R')^\eta, S := (S')^\eta, V := G^{v'+\eta v''}, W := H^{v'+\eta v''}$ and check if (A, C, D, R, S) is valid on $((M, N), (Y, Z), (V, W))$ under ipk;
2. choose $(vk_{\text{ot}}, sk_{\text{ot}}) \leftarrow \mathsf{KeyGen}_{\text{ot}}$ and define $\Sigma \leftarrow \mathsf{Sign}(y, vk_{\text{ot}})$;
3. make commitments $\mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S$ to A, C, D, R, S under ck;
4. run $\mathsf{CEP}(ck, epk, vk_{\text{ot}}, \cdot)$ on Y, Z, Σ; let ψ_Y, ψ_Z, $\vec{\psi}_\Sigma$ denote the outputs;

[4] A simulation-sound non-interactive proof of knowledge of Q and W' such that $e(V', F) = e(G, W')$ and $e(P, F) = e(G, Q)$. (cf. §E.5.2).

[5] The message space for $\mathbf{Sig}$ is the set of DH pairs w.r.t. (G, H'). Since all logarithms of vk_{ot} are known when picking a key, the user can complete the second components of the DH pairs.

5. make a proof ϕ_Y that $(Y, Z) \in \mathcal{DH}$ and proofs ϕ_S and ϕ_Σ of validity of the signatures (A, C, D, R, S) and Σ;
6. set $sig \leftarrow \mathsf{Sign}_{\mathrm{ot}}(sk_{\mathrm{ot}}, (V, W, M, N, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\mathrm{ot}}))$.

The signature on (M, N) is

$$(V, W, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\mathrm{ot}}, sig) \ .$$

Verification. A signature is verified by verifying *sig* under vk_{ot}, checking the proofs ϕ_Y, ϕ_S and ϕ_Σ, and verifying the encryptions and NIZK proofs in ψ_Y, ψ_Z and $\vec{\psi}_\Sigma$.

Remark E.6.1 As mentioned by [Fuc09], there are two possible instantiations of the zero-knowledge proof of knowledge in Step 4 of User: either using bit-by-bit techniques (which makes the protocol round-optimal); or optimizing the amount of data sent by adding 3 rounds using interactive concurrent Schnorr proofs.

Theorem E.6.2 The above scheme is an unforgeable blind signature (in the classical sense) under the DLIN, the DHSDH and the HDL assumption.

The proof of unforgeability is by reduction to unforgeability of Scheme E.5.1, analogously to the proof in [Fuc09]. Note that by additionally extracting y and v' from the proof of knowledge, the simulator can make the special signing queries. The proof of blindness is analogous, too.

Opening of a Transcript ("Signature Tracing"). Given a transcript

$$(Y, Z, \Sigma', \mathbf{c}_M, \mathbf{c}_N, \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, J, \zeta, vk'_{\mathrm{ot}}, sig') \ , \quad v''$$

verify Σ', sig', the proofs ϕ_M and ϕ_ξ and the ciphertexts and proofs in ψ_P, ψ_V and $\vec{\psi}_\xi$. If everything is valid, use $rk = ek$ to open the commitments in ψ_P, ψ_V and $\vec{\psi}_\xi$ to P, V' and ξ respectively and set $V := V'P^{v''} = G^{v'+\eta v''}$.

Return $id_\sigma := V$, $upk = Y$ and $\pi := (V', P, v'', \xi, \Sigma')$. The proof π is verified by checking $V = V'P^{v''}$, verifying ξ on V' and P, and verifying Σ' under Y.

Opening of a Signature ("Identity Tracing"). Given a *valid* signature

$$(V, W, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\mathrm{ot}}, sig) \ ,$$

open the commitments in ψ_Y, ψ_Z and $\vec{\psi}_\Sigma$ using ek and return $upk = Y$ and $\pi = \Sigma$. A proof π is verified by checking if Σ is valid on (V, W) under Y.

E.7 Security Proofs

Theorem E.7.1 The above scheme is a secure fair blind signature scheme (in the model defined in §E.2) under the DLIN, the DHSDH and the HDL assumptions.

Due to space limitation, we sketch the security proofs of all security notions.

Blindness (under DLIN).
In the WI setting of GS proofs, commitments and proofs do not reveal anything—and neither do the ciphertexts. Furthermore, for every M and V, there exist η and v' that explain J. In more detail: we proceed by games, Game 0 being the original game. In Game 1, we use the decryption key for the tag-based encryptions to answer tracing queries. Soundness of the NIZK proofs in the ψ's guarantee that the committed and the encrypted values are the same; the games are thus indistinguishable.

In Game 2, we replace the commitment key *ck* by a WI key (indistinguishable under DLIN). In Game 3, we simulate the NIZK proofs in the ψ's and in Game 4, we replace the ciphertexts in the ψ's by encryptions of 0. Games 3 and 4 are indistinguishable by selective-tag weak CCA security of Kiltz' cryptosystem (which follows from DLIN): by unforgeability of the one-time signature, the adversary cannot query a different transcript (or signature) with the same tag as the target transcript (signature), we can thus answer all tracing queries.

In Game 5, we simulate the zero-knowledge proofs in Step 4. In this game, the adversary's view is the following: $J = (KML^yU^{v'})^{\frac{1}{\eta}}$ and M^*, V^* which are either M and $G^{v'+\eta v''}$ or not. Let small letters denote the logarithms of the respective capital letters. Then for every $m^* = \log M^*, v^* = \log V^*$ there exist η, v' such that $v^* = v' + \eta v''$ and $j = \frac{1}{\eta}(k + m^* + yl + v'u)$, i.e. that make M^*, V^* consistent with J. In Game 5, which is indistinguishable from the original game, the adversary has thus no information on whether a given transcript corresponds to a given signature.

Identity Traceability (under DHSDH+HDL).
An adversary wins if he can produce a set of valid pairs (m_i, σ_i) s.t. either (I) for one of them the tracing returns $\perp$ or the proof does not verify, or (II) a user appears more often in the openings of the signatures than in the openings of the transcripts. By soundness of Groth-Sahai, we can always extract a user public key and a valid signature. If an adversary wins by (II), then we can use him to forge a $\mathbf{Sig}_3$ signature:

Given parameters and a public key for $\mathbf{Sig}_3$, we set up the rest of the parameters for the blind signature. Whenever the adversary queries his Sign oracle, we do the following: use *ek* to extract (M, N) from $(\mathbf{c}_M, \mathbf{c}_N)$, extract η, y and v' from the zero-knowledge proof of knowledge ζ. Choose $v'' \leftarrow \mathbb{Z}_p$ and query $(M, N, y, v' + \eta v'')$ to signing oracle, receive (A, C, D, R, S) and return $(A^{\frac{1}{\eta}}, C, D, R^{\frac{1}{\eta}}, S^{\frac{1}{\eta}}, v'')$. If the adversary wins by outputting a set of different (i.e. with distinct identifiers (V, W)) blind signatures with one user appearing more often than in the transcripts then among the $\mathbf{Sig}_3$ signatures extracted from the blind signatures there must be a forgery.

Identity Non-Frameability (under DLIN+DHSDH+HDL).
Using a successful adversary, we can either forge a signature by the user on vk'_{ot} or a one-time signature (which is secure under DLIN). More precisely, we call an adversary of Type I if it reuses a one-time key from the signatures it received from the User oracle. Since the signature that $\mathcal{A}$ returns must not be contained in *Set*, it is different from the one containing the reused one-time key. The contained one-time signature can thus be returned as a forgery.

An adversary of Type II uses a new one-time key for the returned signature. We use $\mathcal{A}$ to forge a $\mathbf{Sig}$ signature. The simulator is given parameters (H', K, T) and a public key Y for $\mathbf{Sig}$, sets it as one of the honest users' *upk* and queries its signing oracle to simulate the user. Having set $H = G^h$, the simulator can produce $Z = H^y = Y^h$ in the User oracle queries. Since the vk'_{ot} contained $\mathcal{A}$'s output was never queried, we get a valid forgery.

Signature Traceability (under DHSDH+HDL).
If the adversary wins by outputting a message/signature pair with an identifier (V, W) s.t. no transcript opens to it, we can extract a $\mathbf{Sig}_3$ signature on (M, N, Y, Z, V, W) without having ever queried a signature on any $(\cdot, \cdot, \cdot, \cdot, V, W)$. The simulation is done analogously to the proof of identity traceability. If the adversary outputs two different signatures they must have different identifiers; one of the ChkSig calls in the experiment returns thus 0. Note that with overwhelming probability two identifiers from different issuing sessions are different (since v'' is chosen randomly by the experiment *after* the adversary chose v' and η).

Signature Non-Frameability (under DLIN+DHSDH+HDL).
There are two ways for an issuer to "wrongfully" open a transcript: either he opens it to a user (not necessarily honest) and an identifier of a signature which was produced by an honest user in another session; or it opens to an honest user who has not participated in the issuing session.

Framing an honest signature. Suppose the adversary impersonating the issuer manages to produce a new opening of a transcript that leads to an honestly generated signature. We reduce this framing attack to break CDH, whose hardness is implied by DLIN. Let (G, F, V') be a CDH challenge, i.e. we seek to produce $W' := F^{(\log_G V')}$. Set up the parameters of the scheme setting $H = G^h$ and knowing the trapdoor for SSNIZKPoK. In one of the adversary's User oracle calls, choose $\eta \leftarrow \mathbb{Z}_p$ and use V' from the CDH challenge. Simulate the proof of knowledge of W'. Let v'' be the value returned by the adversary, and let $(V := V'P^\eta, W := V^h)$ be the identifier of the resulting signature.

Suppose the adversary produces a proof $(\bar{V}', \bar{P}, \bar{v}'', \bar{\pi}, \bar{\Sigma})$ with $(\bar{V}', \bar{P}) \neq (V', P)$ for the honest identifier (V, W). By simulation soundness of SSNIZKPoK, we can extract $\bar{W}' = F^{(\log_G \bar{V}')}$ and $\bar{Q} = F^{(\log_G \bar{P})}$. From $V'G^{\eta v''} = V = \bar{V}'\bar{P}^{\bar{v}''}$ we get $V' = \bar{V}'\bar{P}^{\bar{v}''}G^{-\eta v''}$; thus $W' = \bar{W}'\bar{Q}^{\bar{v}''}F^{-\eta v''}$ is a CDH solution. If the adversary recycles (V', P) then it must find a new v'' which leads to a V of an honest signature, and thus has to solve a discrete logarithm.

Framing an honest user. Suppose the adversary outputs an opening of a transcript and a proof revealing an honest user that has never participated in that transcript. Analogously to the proof for signature traceability, we can use the adversary to either forge a signature under a user public key or to forge a one-time signature.

E.8 Conclusion

We presented the first practical fair blind signature scheme with a security proof in the standard model. The scheme satisfies a new security model strengthening the one proposed by Hufschmitt and Traoré in 2007. The new scheme is efficient (both keys and signatures consist of a constant number of group elements) and does not rely on any new assumptions. As byproducts, we proposed an extension of Fuchsbauer's automorphic signatures, a one-time signature on group elements, and a simulation-sound non-interactive zero-knowledge proof of knowledge of a Diffie-Hellman solution, all three compatible with the Groth-Sahai methodology.

Acknowledgments

This work was supported by the French ANR 07-TCOM-013-04 PACE Project, the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II, and EADS.

E.A A One-Time Signature on Vectors of Group Elements

Our one-time signature is based on the *simultaneous triple pairing assumption* (STP) stating that the following problem is hard:

> Given random generators $(g_r, h_r, g_s, h_s, g_t, h_t) \in \mathbb{G}^6$, output $(r, s, t) \in \mathbb{G}^3 \setminus \{(1,1,1)\}$ such that
>
> $$e(g_r, r)\, e(g_s, s)\, e(g_t, t) = 1 \qquad e(h_r, r)\, e(h_s, s)\, e(h_t, t) = 1$$

In Groth [Gro09] proves that DLIN implies STP and presents a homomorphic commitment scheme whose binding property is implied by the above assumption. We transform his commitment scheme to a one-time signature scheme analogous to the scheme in [Gro06] based on Pedersen commitments. The signature uses a commitment with an additional trapdoor. The public key is a commitment to 0 and a signature is a trapdoor opening of the commitment to the message.

We give a scheme with message space $\mathbb{G}^n$.

$\mathsf{KeyGen}_{\mathbf{ot}}$ Choose $x_r, y_r, x_s, y_s, x_t, y_t, x_1, y_1, \ldots, x_n, y_n, v, w \leftarrow \mathbb{Z}_p$ such that $x_r y_s \neq x_s y_r$. Define $g_i := g^{x_i}, h_i =: g^{y_i}$ for $i = r, s, t, 1, \ldots, n$, $c = g^v, d = g^w$. Let $\alpha, \beta, \gamma, \delta$ s.t. $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} x_r & x_s \\ y_r & y_s \end{pmatrix}^{-1}$. The public key is

$$(c, d, \vec{g} = (g_r, g_s, g_t, g_1, \ldots, g_n), \vec{h} = (h_r, h_s, h_t, h_1, \ldots, h_n))$$

and the secret key is $(\alpha, \beta, \gamma, \delta, x_t, y_t, x_1, y_1, \ldots, x_n, y_n)$.

$\mathsf{Sign}_{\mathbf{ot}}$ To sign a message $(m_1, \ldots, m_n) \in \mathbb{G}^n$. Choose $t \leftarrow \mathbb{G}$ and set $a := c\, t^{-x_t} \prod m_i^{-x_i}$ and $b := d\, t^{-y_t} \prod m_i^{-y_i}$. Return $(r = a^\alpha b^\beta, s = a^\gamma b^\delta, t)$.

$\mathsf{Verify}_{\mathbf{ot}}$ A signature (r, s, t) is verified on $(m_1, \ldots, m_n)$ by checking

$$e(g_r, r)\, e(g_s, s)\, e(g_t, t) \prod e(g_i, m_i) = e(c, g)$$
$$e(h_r, r)\, e(h_s, s)\, e(h_t, t) \prod e(h_i, m_i) = e(d, g)$$

A signature produced by $\mathsf{Sign}_{\mathrm{ot}}$ is indeed accepted by $\mathsf{Verify}_{\mathrm{ot}}$ since:

$$\begin{aligned}
e(g_r, r)\, e(g_s, s)\, e(g_t, t) \prod e(g_i, m_i) &= e(g_r, a^\alpha b^\beta)\, e(g_s, a^\gamma b^\delta)\, e(g_t, t) \prod e(g_i, m_i) \\
&= e(a^{\alpha x_r + \gamma x_s}, g) e(b^{\beta x_r + \delta x_s}, g)\, e(g_t, t) \prod e(g_i, m_i) \\
&= e(a, g)\, e(g_t, t) \prod e(g_i, m_i) \\
&= e(c\, t^{-x_t} \prod m_i^{-x_i}, g)\, e(g_t, t) \prod e(g_i, m_i) \\
&= e(c, g) e(t^{-x_t}, g)\, e(g_t, t) \prod e(m_i^{-x_i}, g) e(g_i, m_i) \\
&= e(c, g)
\end{aligned}$$

and similarly

$$\begin{aligned}
e(h_r, r)\, e(h_s, s)\, e(h_t, t) \prod e(h_i, m_i) &= e(h_r, a^\alpha b^\beta)\, e(h_s, a^\gamma b^\delta)\, e(h_t, t) \prod e(h_i, m_i) \\
&= e(a^{\alpha y_r + \gamma y_s}, g) e(b^{\beta y_r + \delta y_s}, g)\, e(h_t, t) \prod e(h_i, m_i) \\
&= e(b, g)\, e(h_t, t) \prod e(h_i, m_i) \\
&= e(d, g)
\end{aligned}$$

Assuming STP, the signature is strongly unforgeable under a one-time chosen message attack. Let $(g_r, h_r, g_s, h_s, g_t, h_t)$ be an STP instance. If (g_r, g_s, h_r, h_s) is a Diffie-Hellman (DH) tuple, (i.e., $e(g_r, h_s) = e(g_s, h_r)$), we have an STP solution $(g_s, g_r^{-1}, 1)$, since $e(g_r, g_s)e(g_s, g_r^{-1})e(g_t, 1) = 1$ and $e(h_r, g_s)e(h_s, g_r^{-1})e(h_t, 1) = 1$.

If (g_r, g_s, h_r, h_s) is not a DH-tuple, we choose $\bar{\rho}, \bar{\sigma}, \bar{\tau}, \rho_1, \sigma_1, \tau_1, \ldots, \rho_n, \sigma_n, \tau_n \leftarrow \mathbb{Z}_p$ and set $g_i := g_r^{\rho_i} g_s^{\sigma_i} g_t^{\tau_i}, h_i := h_r^{\rho_i} h_s^{\sigma_i} h_t^{\tau_i}$, for $1 \leq i \leq n$; and $c := g_r^{\bar{\rho}} g_s^{\bar{\sigma}} g_t^{\bar{\tau}}, d := h_r^{\bar{\rho}} h_s^{\bar{\sigma}} h_t^{\bar{\tau}}$. Since (g_r, g_s) and (h_r, h_s) are "linearly independent", all these group elements look random. We give the adversary the public key $(c, d, \vec{g}, \vec{h})$. The signing query for $(m_1, \ldots, m_n)$ is answered by returning $r = g^{\bar{\rho}} \prod m_i^{-\rho_i}$, $s = g^{\bar{\sigma}} \prod m_i^{-\sigma_i}$, $t = g^{\bar{\tau}} \prod m_i^{-\tau_i}$. We have:

$$\begin{aligned} e(g_r, r)\, e(g_s, s)\, e(g_t, t) &= e(g_r, g^{\bar{\rho}} \textstyle\prod m_i^{-\rho_i}) e(g_s, g^{\bar{\sigma}} \prod m_i^{-\sigma_i}) e(g_t, g^{\bar{\tau}} \prod m_i^{-\tau_i}) \\ &= e(g_r^{\bar{\rho}} g_s^{\bar{\sigma}} g_t^{\bar{\tau}}, g) \textstyle\prod (g_r^{-\rho_i} g_s^{-\sigma_i} g_t^{-\tau_i}, m_i) \\ &= e(c, g) \textstyle\prod e(g_i^{-1}, m_i) \end{aligned}$$

and similarly

$$e(h_r, r)\, e(h_s, s)\, e(h_t, t) = e(d, g) \textstyle\prod e(h_i^{-1}, m_i).$$

Thus (r, s, t) is a valid signature for $(m_1, \ldots, m_n)$ and since $\bar{\tau}$ and the τ_i's are perfectly hidden, this looks like a random signature produced by $\mathsf{Sign}_{\mathrm{ot}}$.

Suppose the adversary outputs $(m'_1, \ldots, m'_n, r', s', t') \neq (m_1, \ldots, m_n, r, s, t)$. Dividing the verification relation for each signatures yields:

$$e(g_r, r' r^{-1} \textstyle\prod (m'_i m_i^{-1})^{\rho_i})\, e(g_s, s' s^{-1} \prod (m'_i m_i^{-1})^{\sigma_i})\, e(g_t, t' t^{-1} \prod (m'_i m_i^{-1})^{\tau_i}) = 1$$
$$e(h_r, r' r^{-1} \textstyle\prod (m'_i m_i^{-1})^{\rho_i})\, e(h_s, s' s^{-1} \prod (m'_i m_i^{-1})^{\sigma_i})\, e(h_t, t' t^{-1} \prod (m'_i m_i^{-1})^{\tau_i}) = 1$$

If $(m'_1, \ldots, m'_n) = (m_1, \ldots, m_n)$, then $(r' r^{-1}, s' s^{-1}, t' t^{-1}) \neq (1, 1, 1)$ and these relations provide a solution to the STP problem. Otherwise, if we denote $I \subset \{1, \ldots, n\}$, the set of indices for which $m'_i \neq m_i$ and $n_i := m'_i m_i^{-1}$, the probability that the adversary's output satisfies $r' \prod_{i \in S} n_i^{\rho_i} = r$ is upper-bounded by $1/p$ since the ρ_i's are perfectly hidden. Therefore if $(m'_1, \ldots, m'_n) \neq (m_1, \ldots, m_n)$, we also obtain a solution to the STP problem with overwhelming probability.

$\mathbf{Exp}^{\text{blind-b}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$

$b' \leftarrow \mathcal{A}(pp, ipk, isk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Challenge}_b, \mathsf{User}, \mathsf{TrSig}, \mathsf{TrId})$

return b'

$\mathbf{Exp}^{\text{IdTrac}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$; $Trans \leftarrow \emptyset$

$(m_1, \sigma_1, \ldots, m_n, \sigma_n) \leftarrow \mathcal{A}(pp, ipk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$

for $i = 1 \ldots |Trans|$ do $(upk_i, id_i, \pi_i) \leftarrow \mathsf{TrSig}(pp, rk, ipk, trans_i)$

for $i = 1 \ldots n$ do $(upk'_i, \pi'_i) \leftarrow \mathsf{TrId}(pp, rk, ipk, m_i, \sigma_i)$

if $\exists\, i : upk'_i = \perp$ or $\mathsf{ChkId}(pp, ipk, (m_i, \sigma_i), upk'_i, \pi'_i) = 0$ then return 1

if some upk appears more often in $(upk'_1, \ldots, upk'_n)$ than in $(upk_1, \ldots, upk_{|Trans|})$

then return 1; else return 0

$\mathbf{Exp}^{\text{IdNF}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$

$Set \leftarrow \emptyset$; $HU \leftarrow \emptyset$; $CU \leftarrow \emptyset$

$(upk, m, \sigma, \pi) \leftarrow \mathcal{A}(pp, ipk, isk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$

if $\mathsf{Ver}(pp, ipk, m, \sigma) = 0$ or $\mathsf{ChkId}(pp, ipk, m, \sigma, upk, \pi) = 0$ then return 0

if $(upk, m, \cdot, \sigma) \notin Set$ and $upk \in HU$ then return 1; else return 0

$\mathbf{Exp}^{\text{SigTrac}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$; $Trans \leftarrow \emptyset$

$(m_1, \sigma_1, m_2, \sigma_2) \leftarrow \mathcal{A}(pp, ipk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$

let $Trans = (trans_i)_{i=1}^n$; for $i = 1 \ldots n$ do $(upk_i, id_i, \pi_i) \leftarrow \mathsf{TrSig}(pp, rk, ipk, trans_i)$

if $\mathsf{Ver}(pp, ipk, m_1, \sigma_1) = 1$ and $\forall\, i : \mathsf{ChkSig}(pp, ipk, trans_i, m_1, \sigma_1, upk_i, id_i, \pi_i) = 0$ then return 1

if $(m_1, \sigma_1) \neq (m_2, \sigma_2)$ and $\mathsf{Ver}(pp, ipk, m_1, \sigma_1) = 1$ and $\mathsf{Ver}(pp, ipk, m_2, \sigma_2) = 1$

and $\exists\, i : \mathsf{ChkSig}(pp, ipk, trans_i, m_1, \sigma_1, upk_i, id_i, \pi_i) = \mathsf{ChkSig}(pp, ipk, trans_i, m_2, \sigma_2, upk_i, id_i, \pi_i)) = 1$

then return 1; else return 0

$\mathbf{Exp}^{\text{SigNF}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$

$Set \leftarrow \emptyset$; $HU \leftarrow \emptyset$; $CU \leftarrow \emptyset$

$(trans^*, m^*, \sigma^*, upk^*, id^*_\sigma, \pi^*) \leftarrow \mathcal{A}(pp, ipk, isk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$

let $Set = (upk_i, m_i, trans_i, \sigma_i)_{i=1}^n$

if $\exists\, i : trans^* \neq trans_i$ and $\mathsf{ChkSig}(pp, ipk, trans^*, m_i, \sigma_i, upk^*, id^*_\sigma, \pi^*) = 1$

then return 1

if $(\forall\, i : upk^* = upk_i \Rightarrow trans^* \neq trans_i$ and $\mathsf{ChkSig}(\ldots, trans^*, m^*, \sigma^*, upk^*, id^*_\sigma, \pi^*) = 1$

then return 1; else return 0

Figure E.1: Security experiments for fair blind signatures

Appendix F

Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model

CANS 2009
[LV09] with B. Libert

Abstract : Group signatures allow users to anonymously sign messages in the name of a group. Membership revocation has always been a critical issue in such systems. In 2004, Boneh and Shacham formalized the concept of group signatures with *verifier-local revocation* where revocation messages are only sent to signature verifiers (as opposed to both signers and verifiers). This paper presents an efficient verifier-local revocation group signature (VLR-GS) providing *backward unlinkability* (*i.e.* previously issued signatures remain anonymous even after the signer's revocation) with a security proof in the standard model (*i.e.* without resorting to the random oracle heuristic).

F.1 Introduction

The *group signature* primitive, as introduced by Chaum and van Heyst in 1991 [Cv91], allows members of a group to sign messages, while hiding their identity within a population group members administered by a group manager. At the same time, it must be possible for a tracing authority holding some trapdoor information to "open" signatures and find out which group members are their originator. A major issue in group signatures is the revocation of users whose membership should be cancelled: disabling the signing capability of misbehaving members (or honest users who intentionally leave the group) without affecting remaining members happens to be a highly non-trivial problem. In 2004, Boneh and Shacham [BS04] formalized the concept of group signatures with *verifier-local revocation* where revocation messages are only sent to signature verifiers (as opposed to both signers and verifiers). This paper describes the first efficient verifier-local revocation group signature scheme providing *backward unlinkability* (*i.e.*, previously issued signatures remain anonymous even after the signer's revocation) whose proof of security does not hinge upon the random oracle heuristic.

F.1.1 Related Work

Group signatures. Many group signatures were proposed in the nineties, the first provably coalition-resistant proposal being the famous ACJT scheme [ACJT00] proposed by Ateniese, Camenisch, Joye and Tsudik in 2000. The last few years saw the appearance of new constructions using bilinear maps [BBS04, NSN04, FI05, DP06]. Among these, the Boneh-Boyen-Shacham scheme [BBS04] was the first one to offer signatures shorter than 200 bytes using the *Strong Diffie-Hellman assumption* [BB04b]. Its security was analyzed using random oracles [BR93] in the model of Bellare, Micciancio and Warinschi [BMW03] (BMW) which captures all the requirements of group signatures in three well-defined properties.

The BMW model, which assumes static groups where no new member can be introduced after the setup phase, was independently extended by Kiayias and Yung [KY04] and Bellare-Shi-Zhang [BSZ05] to a dynamic setting. In these models (that are very close to each other), efficient pairing-based schemes were put forth by Nguyen and Safavi-Naini [NSN04], Furukawa and Imai [FI05] and, later on, by Delerablée and Pointcheval [DP06]. In dynamically growing groups, Ateniese *et al.* [ACHdM05] also proposed a construction without random oracles offering a competitive efficiency at the expense of a security resting on interactive assumptions that are not efficiently falsifiable [Nao03]. Another standard model proposal was put forth (and subsequently improved [BW07]) by Boyen-Waters [BW06b] in the static model from [BMW03] under more classical assumptions. Groth [Gro06] described a scheme with constant-size signatures without random oracles in the dynamic model [BSZ05] but signatures were still too long for practical use. Later on, Groth showed [Gro07] a fairly practical random-oracle-free group signature with signature length smaller than 2 kB and full anonymity (*i.e.*, anonymity in a model where the adversary is allowed to open anonymous signatures at will) in the model of [BSZ05].

Verifier-local revocation. Membership revocation has always been a critical issue in group signatures. The simplest solution is to generate a new group public key and provide unrevoked signers with a new signing key, which implies the group master to send a secret message to each individual signer as well as to broadcast a public message to verifiers. In some settings, it may not be convenient to send a new secret to signers after their inclusion in the group. In *verifier-local revocation group signatures* (VLR-GS), originally suggested in [Bri03] and formalized in [BS04], revocation messages are only sent to verifiers (making the group public key and the signing procedure independent of which and how many members were excluded). The group manager maintains a (periodically updated) revocation list (RL) which is used by all verifiers to perform the revocation test and make sure that signatures were not produced by a revoked member.

The RL contains a token for each revoked user. The verification algorithm accepts all signatures issued by unrevoked users and reveals no information about which unrevoked user issued the signature. However, if a user is revoked, his signatures are no longer accepted. It follows that signatures from a revoked member become linkable: to test that two signatures emanate from the same revoked user, one can simply verify signatures once using the RL before the alleged signer's revocation and once using the post-revocation RL. As a result, users who deliberately leave the group inevitably lose their privacy.

The property of *backward unlinkability*, first introduced in [Son01] in the context of key-evolving group signatures, ensures that signatures that were generated by a revoked member *before* his revocation remain anonymous and unlinkable. This property is useful when members who voluntarily leave the group wish to retain a certain level of privacy. When users' private keys get stolen, preserving the anonymity of their prior signatures is also definitely desirable.

Boneh and Shacham [BS04] proposed a VLR group signature using bilinear maps in a model inspired from [BMW03]. In [NF05], Nakanishi and Funabiki extended Boneh-Shacham group signatures and devised a scheme providing backward unlinkability. They proved the anonymity

of their construction under the Decision Bilinear Diffie-Hellman assumption [BF01]. In [NF06], the same authors suggested another backward-unlinkable scheme with shorter signatures. Other pairing-based VLR-GS constructions were put forth in [ZL06a, ZL06b]

Traceable signatures [KTY04], that also have pairing-based realizations [NSN04, CPY06], can be seen as extensions of VLR-GS schemes as they also admit an implicit tracing mechanism. They provide additional useful properties such as the ability for signers to claim (and prove) the authorship of anonymously generated signatures or the ability for the group manager to reveal a trapdoor allowing to publicly trace all signatures created by a given user. This primitive was recently implemented in the standard model [LY09]. However, it currently does not provide a way to trace users' signatures per period: once the tracing trapdoor of some group member is revealed, all signatures created by that member become linkable. In some situations, it may be desirable to obtain a fine-grained traceability and only trace signatures that were issued in specific periods. The problem of VLR-GS schemes with backward unlinkability can be seen as the one of tracing some user's signatures from a given period onwards while preserving the anonymity and the unlinkability of that user's signatures for earlier periods. The solution described in this paper readily extends to retain the anonymity of signatures produced during past *and* future periods.

F.1.2 Contribution of the paper.

All known constructions of group signatures with verifier local revocation (with or without backward unlinkability) make use of the Fiat-Shamir paradigm [FS86] and thus rely on the random oracle methodology [BR93], which is known not to provide more than heuristic arguments in terms of security. Failures of the random oracle model were indeed reported in several papers such as [CGH04, GK03]. When first analyzed in the random oracle model, cryptographic primitives thus deserve further efforts towards securely instantiating them without appealing to the random oracle idealization.

The contribution of this paper is to describe a new VLR-GS scheme with backward unlinkability in the standard model. Recently, Groth and Sahai [GS08] described powerful non-interactive proof systems allowing to prove that a number of committed variables satisfy certain algebraic relations. Their techniques notably proved useful to design standard model group signatures featuring constant signature size [BW07, Gro06, Gro07].

Extending the aforementioned constructions to obtain VLR-GS schemes with backward unlinkability is not straightforward. The approach used in [NF06], which can be traced back to Boneh-Shacham [BS04], inherently requires to use programmable random oracles, the behavior of which currently seems impossible to emulate in the standard model (even with the techniques developed in [HK08]). Another approach used in [NF05] looks more promising as it permits traceability with backward unlinkablity without introducing additional random oracles. This technique, however, does not interact with the Groth-Sahai toolbox in a straightforward manner as it typically requires non-interactive zero-knowledge (NIZK) proofs for what Groth and Sahai called *pairing product equations*. The problem that we face is that proving the required anonymity property of VLR-GS schemes entails to simulate a NIZK proof for such a pairing-product equation at some step of the reduction. As pointed out in [GS08], such non-interactive proofs are only known to be simulatable in NIZK under specific circumstances that are not met if we try to directly apply the technique of [NF05].

To address the above technical difficulty, we use the same revocation mechanism as [NF05] but use a slightly stronger (but still falsifiable [Nao03]) assumption in the proof of anonymity: while Nakanishi and Funabiki rely the Decision Bilinear Diffie-Hellman assumption, we rest on the hardness of the so-called Decision Tripartite Diffie-Hellman problem, which is to distinguish g^{abc} from random given (g, g^a, g^b, g^c). Our contribution can be summarized as showing that the

implicit tracing mechanism of [NF05] can be safely applied to the Boyen-Waters group signature [BW07] to make it backward-unlinkably revocable. This property comes at the expense of a quite moderate increase of signature sizes w.r.t. [BW07]. The main price to pay is actually to use a slightly stronger assumption than in [NF05] in the security proof.

F.2 Preliminaries

F.2.1 Verifier-Local Revocation Group Signatures

This section presents the model of VLR group signatures with backward unlinkability proposed in [NF05] which extends the Boneh-Shacham model [BS04] of VLR group signatures.

Definition F.2.1 A VLR group signature scheme with backward unlinkability consists of the following algorithms:

Keygen(λ, N, T): is a randomized algorithm taking as input a security parameter $\lambda \in \mathbb{N}$ and integers $N, T \in \mathbb{N}$ indicating the number of group members and the number of time periods, respectively.

Its output consists of a group public key gpk, a N-vector of group members' secret keys $\mathsf{gsk} = (\mathsf{gsk}[1], \ldots, \mathsf{gsk}[N])$ and a $(N \times T)$-vector of revocation tokens

$$\mathsf{grt} = (\mathsf{grt}[1][1], \ldots, \mathsf{grt}[N][T]),$$

where $\mathsf{grt}[i][j]$ indicates the token of member i at time interval j.

Sign$(\mathsf{gpk}, \mathsf{gsk}[i], j, M)$: is a possibly randomized algorithm taking as input, the group public key gpk, the current time interval j, a group member's secret key $\mathsf{gsk}[i]$ and a message $M \in \{0,1\}^*$. It outputs a group signature σ.

Verify$(\mathsf{gpk}, j, RL_j, \sigma, M)$: is a deterministic algorithm taking as input gpk, the period number j, a set of revocation tokens RL_j for period j, a signature σ, and the message M. It outputs either "`valid`" or "`invalid`". The former output indicates that σ is a correct signature on M at interval j w.r.t. gpk, and that the signer is not revoked at interval j.

For all $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grt}) = \mathbf{Keygen}(\lambda, N, T)$, all $j \in \{1, \ldots, T\}$, all RL_j, all $i \in \{1, \ldots, N\}$ and any message $M \in \{0,1\}^*$, it is required that if $\mathsf{grt}[i][j] \notin RL_j$ then:

$$\mathbf{Verify}(\mathsf{gpk}, j, RL_j, \mathbf{Sign}(\mathsf{gpk}, \mathsf{gsk}[i], j, M), M) = \text{"}\texttt{valid}\text{"}.$$

Remark F.2.2 As mentioned in [BS04], any such group signature scheme has an associated *implicit tracing* algorithm that allows tracing a signature to the group member who generated it using the vector grt as the tracing key: on input a valid message-signature pair (M, σ) for period j, the opener can determine which user was the author of σ by successively executing the verification algorithm on (M, σ) using the vector of revocation tokens (*i.e.*, with $RL_j = \{\mathsf{grt}[i][j]\}_{i \in \{1,\ldots,N\}}$) and outputting the first index $i \in \{1, \ldots, N\}$ for which the verification algorithm returns "`invalid`" whereas verifying the same pair (M, σ) with $RL_j = \emptyset$ yields the answer "`valid`".

From a security standpoint, VLR group signatures with backward unlinkability should satisfy the following properties:

Definition F.2.3 A VLR-GS with backward unlinkability has the **traceability** property if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has non-negligible advantage in the following game.

1. The challenger $\mathcal{C}$ runs the setup algorithm to produce a group public key gpk, a group master secret gsk and a vector grt of revocation tokens. It also defines a set of corrupt users U which is initially empty. The adversary $\mathcal{A}$ is provided with gpk and grt while $\mathcal{C}$ keeps gsk to itself.

2. $\mathcal{A}$ can make a number of invocations to the following oracles:

   Signing oracle: on input of a message M, an index $i \in \{1, \ldots, N\}$ and a period number j, this oracle responds with a signature σ generated on behalf of member i for period j.

   Corruption oracle: given an index $i \in \{1, \ldots, N\}$, this oracle reveals the private key $\mathsf{gsk}[i]$ of member i which is included in the set U.

3. $\mathcal{A}$ eventually comes up with a signature $\sigma^\star$ on a message $M^\star$, a period number $j^\star$ and a set of revocation tokens $RL^\star_{j^\star}$.

The adversary $\mathcal{A}$ is declared successful if

- $\mathbf{Verify}(\mathsf{gpk}, j^\star, RL^\star_{j^\star}, \sigma^\star, M^\star) =$ "`valid`".
- The execution of the implicit tracing algorithm on input of revocation tokens

  $$(\mathsf{grt}[1][j^\star], \ldots, \mathsf{grt}[N][j^\star]),$$

  ends up in one of the following ways:

  - $\sigma^\star$ traces to a member outside the coalition $U \backslash RL^\star_{j^\star}$ that did not sign $M^\star$ during period $j^\star$
  - the tracing fails.

$\mathcal{A}$'s advantage in breaking traceability is measured as

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{A}}(k) \;:=\; \Pr[\mathcal{A} \text{ is successful}],$$

where the probability is taken over the coin tosses of $\mathcal{A}$ and the challenger.

This definition slightly weakens the original one [NF05] that captures the strong unforgeability requirement (*i.e.*, the message-signature pair $(M^\star, \sigma^\star)$ must be different from that of any signing query during period $j^\star$). Due to the use of publicly randomizable non-interactive witness indistinguishable proofs, we need to settle for the usual flavor of unforgeability according to which the message $M^\star$ must not have been queried for signature during the target period $j^\star$.

Definition F.2.4 A VLR-GS with backward unlinkability provides **BU-anonymity** if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game.

1. The challenger $\mathcal{C}$ runs $\mathbf{Keygen}(\lambda, n, T)$ to produce a group public key gpk, a master secret gsk and a vector grt of revocation tokens. The adversary $\mathcal{A}$ is given gpk but is denied access to grt and gsk.

2. At the beginning of each period, $\mathcal{C}$ increments a counter j and notifies $\mathcal{A}$ about it. During the current time interval j, $\mathcal{A}$ can adaptively invoke the following oracles:

   Signing oracle: on input of a message M and an index $i \in \{1, \ldots, n\}$, this oracle outputs a signature σ generated for member i and period j.

   Corruption oracle: for an adversarially-chosen $i \in \{1, \ldots, n\}$, this oracle reveals member i's private key $\mathsf{gsk}[i]$.

   Revocation oracle: given $i \in \{1, \ldots, n\}$, this oracle outputs member i's revocation token for the current period j.

3. At some period $j^\star \in \{1, \ldots, T\}$, $\mathcal{A}$ comes up with a message M and two distinct user indices $i_0, i_1 \in \{1, \ldots, n\}$ such that neither i_0 or i_1 has been corrupt. Moreover, they cannot have been revoked before or during period $j^\star$. At this stage, $\mathcal{C}$ flips a fair coin $d^\star \xleftarrow{R} \{0,1\}$ and generates a signature $\sigma^\star$ on M on behalf of user $i_{d^\star}$ which is sent as a challenge to $\mathcal{A}$.

4. $\mathcal{A}$ is granted further oracle accesses as in phase 2. Of course, she may not query the private key of members i_0, i_1 at any time. On the other hand, she may obtain their revocation tokens for time intervals after $j^\star$.

5. Eventually, $\mathcal{A}$ outputs $d' \in \{0,1\}$ and wins if $d' = d^\star$.

The advantage of $\mathcal{A}$ in breaking BU-anonymity is defined as $\mathbf{Adv}_{\mathcal{A}}^{\text{bu-anon}}(k) := |\Pr[d' = d^\star] - 1/2|$, where the probability is taken over all coin tosses.

F.2.2 Bilinear Maps and Complexity Assumptions

Bilinear groups. Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear groups* if there is an efficiently computable mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:

1. $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In such groups, we will need three non-interactive (and thus falsifiable [Nao03]) complexity assumptions.

Definition F.2.5 In a group $\mathbb{G} = \langle g \rangle$ of prime order $p > 2^\lambda$, the **Decision Linear Problem** (DLIN) is to distinguish the distributions $(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g, g^a, g^b, g^{ac}, g^{bd}, g^z)$, with $a, b, c, d \xleftarrow{R} \mathbb{Z}_p^*$, $z \xleftarrow{R} \mathbb{Z}_p^*$. The **Decision Linear Assumption** posits that, for any PPT distinguisher $\mathcal{D}$,

$$\mathbf{Adv}_{\mathbb{G},\mathcal{D}}^{\text{DLIN}}(\lambda) = |\Pr[\mathcal{D}(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d}) = 1 | a, b, c, d \xleftarrow{R} \mathbb{Z}_p^*] - \Pr[\mathcal{D}(g, g^a, g^b, g^{ac}, g^{bd}, g^z) = 1 | a, b, c, d \xleftarrow{R} \mathbb{Z}_p^*,\ z \xleftarrow{R} \mathbb{Z}_p^*]| \in \mathsf{negl}(\lambda).$$

This problem amounts to deciding whether vectors $\vec{g_1} = (g^a, 1, g)$, $\vec{g_2} = (1, g^b, g)$ and $\vec{g_3}$ are linearly dependent or not. It has been used [GS08] to construct efficient non-interactive proof systems.

We also rely on a variant, introduced by Boyen and Waters [BW07], of the Strong Diffie-Hellman assumption [BB04b].

Definition F.2.6 [[BW07]] In a group $\mathbb{G}$ of prime order p, the **ℓ-Hidden Strong Diffie-Hellman problem** (ℓ-HSDH) is, given elements $(g, \Omega = g^{\omega}, u) \xleftarrow{R} \mathbb{G}^3$ and ℓ distinct triples $(g^{1/(\omega+s_i)}, g^{s_i}, u^{s_i})$ with $s_1, \ldots, s_\ell \xleftarrow{R} \mathbb{Z}_p^*$, to find another triple $(g^{1/(\omega+s)}, g^s, u^s)$ such that $s \neq s_i$ for $i \in \{1, \ldots, \ell\}$.

We also rely on the following intractability assumption suggested for the first time in [BF01, Section 8].

Definition F.2.7 In a prime order group $\mathbb{G}$, the **Decision Tripartite Diffie-Hellman Assumption** (DTDH) is the infeasibility of deciding if $\eta = g^{abc}$ on input of (g, g^a, g^b, g^c, η), where $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$. The advantage function $\mathbf{Adv}^{\mathrm{DTDH}}_{\mathbb{G},\mathcal{D}}(\lambda)$ of any PPT distinguisher $\mathcal{D}$ is defined analogously to the DLIN case.

The above assumption is a bit stronger than the widely accepted Decision Bilinear Diffie-Hellman assumption according to which the distributions

$$\{(g, g^a, g^b, g^c, e(g,g)^{abc}) | a, b, c, \xleftarrow{R} \mathbb{Z}_p\} \text{ and } \{(g, g^a, g^b, g^c, e(g,g)^{z}) | a, b, c, z \xleftarrow{R} \mathbb{Z}_p\}$$

are computationally indistinguishable. Yet, the DTDH problem is still believed to be hard in groups with a bilinear map where the DDH problem is easy.

F.2.3 Groth-Sahai Proof Systems

In the following notations, for equal-dimension vectors or matrices A and B containing group elements, $A \odot B$ stands for their entry-wise product (*i.e.* it denotes their Hadamard product).

When based on the DLIN assumption, the Groth-Sahai (GS) proof systems [GS08] use a common reference string comprising vectors $\vec{g_1}, \vec{g_2}, \vec{g_3} \in \mathbb{G}^3$, where $\vec{g_1} = (g_1, 1, g)$, $\vec{g_2} = (1, g_2, g)$ for some $g_1, g_2 \in \mathbb{G}$. To commit to group elements $X \in \mathbb{G}$, one sets $\vec{C} = (1, 1, X) \odot \vec{g_1}^r \odot \vec{g_2}^s \odot \vec{g_3}^t$ with $r, s, t \xleftarrow{R} \mathbb{Z}_p^*$. When the proof system is configured to give perfectly sound proofs, $\vec{g_3}$ is chosen as $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ with $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$. Commitments $\vec{C} = (g_1^{r+\xi_1 t}, g_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ are then Boneh-Boyen-Shacham (C:BonBoySha04) ciphertexts that can be decrypted using $\alpha_1 = \log_g(g_1)$, $\alpha_2 = \log_g(g_2)$. In the witness indistinguishability (WI) setting, vectors $\vec{g_1}, \vec{g_2}, \vec{g_3}$ are linearly independent and $\vec{C}$ is a perfectly hiding commitment. Under the DLIN assumption, the two kinds of CRS are computationally indistinguishable.

To commit to a scalar $x \in \mathbb{Z}_p$, one computes $\vec{C} = \vec{\varphi}^x \odot \vec{g_1}^r \odot \vec{g_2}^s$, with $r, s \xleftarrow{R} \mathbb{Z}_p^*$, using a CRS comprising vectors $\vec{\varphi}, \vec{g_1}, \vec{g_2}$. In the soundness setting $\vec{\varphi}, \vec{g_1}, \vec{g_2}$ are linearly independent (typically $\vec{\varphi} = \vec{g_3} \odot (1, 1, g)$ where $\vec{\varphi} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$) whereas, in the WI setting, choosing $\vec{\varphi} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ gives a perfectly hiding commitment since $\vec{C}$ is always a BBS encryption of $1_{\mathbb{G}}$.

To prove that committed variables satisfy a set of relations, the GS techniques replace variables by commitments in each relation. The whole proof consists of one commitment per variable and one proof element (made of a constant number of group elements) per relation.

Such proofs are easily obtained for pairing-product relations, which are of the type

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^{n} \cdot \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T, \tag{F.1}$$

for committed variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and public constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{G}$, for $i, j \in \{1, \ldots, n\}$. Efficient proofs also exist for multi-exponentiation equations

$$\prod_{i=1}^{m} \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^{m} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{y_i \gamma_{ij}} = T, \tag{F.2}$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$, $y_1, \ldots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathbb{G}$, $b_1, \ldots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{G}$, for $i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$.

In both cases, proofs for quadratic equations cost 9 group elements. Linear pairing-product equations (when $a_{ij} = 0$ for all i, j) take 3 group elements each. Linear multi-exponentiation equations of the type $\prod_{j=1}^{n} \mathcal{X}_j^{b_j} = T$ (resp. $\prod_{i=1}^{m} \mathcal{A}_i^{y_i} = T$) demand 3 (resp. 2) group elements.

Multi-exponentiation equations admit zero-knowledge proofs at no additional cost. On a simulated CRS (prepared for the WI setting), a trapdoor makes it is possible to simulate proofs without knowing witnesses and simulated proofs are identically distributed to real proofs.

On the other hand, pairing-product equations are not known to always have zero-knowledge proofs. Proving relations of the type (F.1) in NIZK usually comes at some expense since auxiliary variables have to be introduced and proof sizes are not necessarily independent of the number of variables. If $t_T = 1_{\mathbb{G}_T}$ in relation (F.1), the NIZK simulator can always use $\mathcal{X}_1 = \cdots = \mathcal{X}_n = 1_{\mathbb{G}}$ as witnesses. If t_T equals $\prod_{j=1}^{n'} e(g_j, h_j)$ for known group elements $g_1, \ldots, g_{n'}, h_1, \ldots, h_{n'} \in \mathbb{G}$, the simulator can prove that

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^{n} \cdot \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = \prod_{j=1}^{n'} e(g_j, \mathcal{Y}_j) \tag{F.3}$$

and that introduced variables $\mathcal{Y}_1, \ldots, \mathcal{Y}_{n'}$ satisfy the linear equations $\mathcal{Y}_j = h_j$ for $j \in \{1, \ldots, n'\}$. Since linear equations are known to have NIZK proofs and the proof of relation (F.3) can be simulated using witnesses $\mathcal{X}_1 = \cdots = \mathcal{X}_n = \mathcal{Y}_1 = \cdots = \mathcal{Y}_{n'} = 1_{\mathbb{G}}$. When t_T is an arbitrary element of $\mathbb{G}_T$, pairing-product equations are currently not known to have NIZK proofs at all.

F.3 A Scheme in the Standard Model

F.3.1 Description of the scheme

In notations hereafter, it will be useful to define the coordinate-wise pairing $E : \mathbb{G} \times \mathbb{G}^3 \to \mathbb{G}_T^3$ such that, for any $h \in \mathbb{G}$ and any vector $\vec{g} = (g_1, g_2, g_3) \in \mathbb{G}^3$,

$$E(h, \vec{g}) = \big(e(h, g_1), e(h, g_2), e(h, g_3)\big).$$

As in [GS08], we will also make use of a symmetric bilinear map $F : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T$ defined in such a way that, for any vectors $\vec{X} = (X_1, X_2, X_3) \in \mathbb{G}^3$ and $\vec{Y} = (Y_1, Y_2, Y_3) \in \mathbb{G}^3$, we have $F(\vec{X}, \vec{Y}) = \tilde{F}(\vec{X}, \vec{Y})^{1/2} \cdot \tilde{F}(\vec{Y}, \vec{X})^{1/2}$, where $\tilde{F} : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T^9$ is a non-commutative bilinear mapping that sends $(\vec{X}, \vec{Y})$ onto the matrix $\tilde{F}(\vec{X}, \vec{Y})$ of entry-wise pairings (*i.e.*, containing $e(X_i, Y_j)$ in its entry (i, j)).

Also, for any $z \in \mathbb{G}_T$, $\iota_T(z)$ denotes the 3×3 matrix containing z in position $(3, 3)$ and 1 everywhere else. For group elements $X \in \mathbb{G}$, the notation $\iota(X)$ will denote the vector $(1, 1, X) \in \mathbb{G}^3$.

The group manager holds a public key $(g, \Omega = g^\omega, A = e(g, g)^\alpha, u)$, where (α, γ) is the private key. As in the Boyen-Waters construction [BW07], group members' private keys consist of triples $(K_1, K_2, K_3) = ((g^\alpha)^{1/(\omega + s_i)}, g^{s_i}, u^{s_i})$, where s_i uniquely identifies the group member. Messages can be signed by creating tuples $(S_1, S_2, S_3, S_4) = (K_1, K_2, K_3 \cdot F(m)^r, g^r)$, where r is a random exponent and $F : \{0, 1\}^* \to \mathbb{G}$ is a Waters-like hash function [Wat05].

The revocation mechanism of [NF05] consists in introducing a vector $(h_1, \ldots, h_T)$ of group elements, where T is the number of time periods, that allow to form revocation tokens for each user: the revocation token of user i for period j is obtained as $\mathsf{grt}[i][j] = h_j^{s_i}$. When user i must be revoked at stage j, the group manager can simply add $\mathsf{grt}[i][j]$ to the revocation list RL_j of period j. When user i signs a message during stage j, he is required to include a pair

$(T_1, T_2) = \left(g^{\delta}, e(h_j, g^{s_i})^{\delta}\right)$ in the signature and append a proof that $(g, T_1 = g^{\delta}, K_2 = g^{s_i}, h_j, T_2)$ satisfy the forementioned relation and that T_2 is indeed the "Bilinear Diffie-Hellman value" $e(h_j, g^{s_i})^{\delta}$ associated with (g, T_1, K_2, h_j).

Keygen(λ, N, T)**:** for security parameters λ and $n \in \mathsf{poly}(\lambda)$, choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^{\lambda}$, with $g, h_1, \ldots, h_T, u \xleftarrow{R} \mathbb{G}$. Select $\alpha, \omega \xleftarrow{R} \mathbb{Z}_p^*$ and set $A = e(g,g)^{\alpha}$, $\Omega = g^{\omega}$. Select $\overline{v} = (v_0, v_1, \ldots, v_n) \xleftarrow{R} \mathbb{G}^{n+1}$. Choose vectors $\mathbf{g} = (\vec{g_1}, \vec{g_2}, \vec{g_3})$ such that $\vec{g_1} = (g_1, 1, g) \in \mathbb{G}^3$, $\vec{g_2} = (1, g_2, g) \in \mathbb{G}^3$, and $\vec{g_3} = \vec{g_1}^{\xi_1} \cdot \vec{g_2}^{\xi_2}$, with $g_1 = g^{\alpha_1}, g_2 = g^{\alpha_2}$ and $\alpha_1, \alpha_2 \xleftarrow{R} \mathbb{Z}_p^*$, $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p$. Finally, select a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^n$. The group public key is defined to be

$$\mathsf{gpk} := \Big(g,\ h_1, \ldots,\ h_T,\ A = e(g,g)^{\alpha},\ \Omega = g^{\omega},\ u,\ \overline{v},\ \mathbf{g},\ H\Big)$$

while the group manager's private key is $(\alpha, \omega, \alpha_1, \alpha_2)$. User i is assigned the group signing key $\mathsf{gsk}[i] = (K_1, K_2, K_3) = \left((g^{\alpha})^{\frac{1}{\omega + s_i}}, g^{s_i}, u^{s_i}\right)$ and his revocation token for period $j \in \{1, \ldots, T\}$ is defined as $\mathsf{grt}[i][j] := h_j^{s_i}$.

Sign$(\mathsf{gpk}, \mathsf{gsk}[i], j, M)$**:** given $\mathsf{gsk}[i] = (K_1, K_2, K_3) = \left((g^{\alpha})^{\frac{1}{\omega + s_i}}, g^{s_i}, u^{s_i}\right)$, to sign a message M during period j, the signer $\mathcal{U}_i$ first computes a hash value $m = m_1 \ldots m_n = H(j||M) \in \{0,1\}^n$ and conducts the following steps.

1. Choose $\delta, r \xleftarrow{R} \mathbb{Z}_p^*$ and first compute

$$T_1 = g^{\delta} \qquad\qquad T_2 = e(h_j, K_2)^{\delta} \tag{F.4}$$

as well as

$$\theta_1 = K_1 = (g^{\alpha})^{1/(\omega + s_i)} \tag{F.5}$$
$$\theta_2 = K_2 = g^{s_i} \tag{F.6}$$
$$\theta_3 = K_3 \cdot F(m)^r = u^{s_i} \cdot F(m)^r \tag{F.7}$$
$$\theta_4 = g^r \tag{F.8}$$
$$\theta_5 = h_j^{\delta}, \tag{F.9}$$

where $F(m) = v_0 \cdot \prod_{k=1}^{n} v_k^{m_k}$.

2. Commit to group elements θ_ℓ, for $\ell \in \{1, \ldots, 5\}$. For $\ell \in \{1, \ldots, 5\}$, choose $r_\ell, s_\ell, t_\ell \xleftarrow{R} \mathbb{Z}_p^*$ and set $\vec{\sigma_\ell} = (1, 1, \theta_\ell) \cdot \vec{g_1}^{r_\ell} \cdot \vec{g_2}^{s_\ell} \cdot \vec{g_3}^{t_\ell}$.

3. Give NIWI proofs that committed variables $\theta_1, \ldots, \theta_4$ satisfy

$$e(\theta_1, \Omega \cdot \theta_2) = A \tag{F.10}$$
$$e(\theta_3, g) = e(u, \theta_2) \cdot e(F(m), \theta_4) \tag{F.11}$$

Relation (F.10) is a quadratic pairing product equation (in the Groth-Sahai terminology) over variables θ_1, θ_2. Such a relation requires a proof consisting of 9 group elements that we denote by $\pi_1 = (\vec{\pi}_{1,1}, \vec{\pi}_{1,2}, \vec{\pi}_{1,3})$. Relation (F.11) is a linear pairing product equation over the variables $\theta_2, \theta_3, \theta_4$. The corresponding proof, that we denote by $\pi_2 = (\pi_{2,1}, \pi_{2,2}, \pi_{2,3}) \in \mathbb{G}^3$, consists of 3 group elements.

5. Give NIZK proofs that committed variables θ_2 and θ_5 satisfy

$$T_2 = e(\theta_2, \theta_5) \tag{F.12}$$

$$e(h_j, T_1) = e(g, \theta_5) \tag{F.13}$$

These are two linear pairing product equations over the variables θ_2 and θ_5 and proving them in NIZK requires to introduce an auxiliary variable θ_6. Proving (F.13) is achieved by proving in NIZK that $e(\theta_6, T_1) = e(g, \theta_5)$ and $\theta_6 = h_j$. The proof for (F.13) thus comprises an auxiliary commitment $\vec{\sigma_6} = \iota(h_j) \odot \vec{g_1}^{r_6} \odot \vec{g_2}^{s_6} \odot \vec{g_3}^{t_6}$ to $\theta_6 = h_j$ and proofs that relations

$$e(\theta_6, T_1) = e(g, \theta_5) \tag{F.14}$$

$$e(\theta_6, g) = e(h_j, g) \tag{F.15}$$

are simultaneously satisfied. These relations are all pairing-product equations. Relation (F.12) is quadratic and costs 9 group elements to prove. We will call this proofs $\pi_3 = (\vec{\pi}_{3,1}, \vec{\pi}_{3,2}, \vec{\pi}_{3,3})$. Relations (F.14)-(F.15) are linear and only require 3 group elements each. The corresponding proofs are denoted by $\pi_4 = (\pi_{4,1}, \pi_{4,2}, \pi_{4,3})$ and $\pi_5 = (\pi_{5,1}, \pi_{5,2}, \pi_{5,3})$.

The signature consists of $\sigma = (T_1, T_2, \vec{\sigma_1}, \ldots, \vec{\sigma_6}, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$.

Verify$(j, M, \sigma, \mathsf{gpk}, RL_j)$**:** parse σ as $(T_1, T_2, \vec{\sigma_1}, \ldots, \vec{\sigma_6}, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$ and return "`valid`" if and only if all proof are valid and σ passes the revocation test:

1. We abstracted away the construction of proof elements $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ for clarity. To explain to proof of anonymity, it will be useful to outline what verification equations look like: namely, $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ must satisfy

   1) $F(\vec{\sigma_1}, \iota(\Omega) \cdot \vec{\sigma_2}) = \iota_T(A) \odot F(\vec{g_1}, \vec{\pi}_{1,1}) \odot F(\vec{g_2}, \vec{\pi}_{1,2}) \odot F(\vec{g_3}, \vec{\pi}_{1,3})$
   2) $E(g, \vec{\sigma_3}) = E(u, \vec{\sigma_2}) \odot E(F(m), \vec{\sigma_4}) \odot E(\pi_{2,1}, \vec{g_1}) \odot E(\pi_{2,2}, \vec{g_2}) \odot E(\pi_{2,3}, \vec{g_3})$
   3) $F(\vec{\sigma_2}, \vec{\sigma_5}) = F(\iota(T_2)) \odot F(\vec{\pi}_{3,1}, \vec{g_1}) \odot F(\vec{\pi}_{3,2}, \vec{g_2}) \odot F(\vec{\pi}_{3,3}, \vec{g_3})$
   4) $E(T_1, \vec{\sigma_6}) = E(\iota(g), \vec{\sigma_5}) \odot E(\pi_{4,1}, \vec{g_1}) \odot E(\pi_{4,2}, \vec{g_2}) \odot E(\pi_{4,3}, \vec{g_3})$
   5) $E(g, \vec{\sigma_6}) = E(h_j, \iota(g)) \odot E(\pi_{5,1}, \vec{g_1}) \odot E(\pi_{5,2}, \vec{g_2}) \odot E(\pi_{5,3}, \vec{g_3})$

2. The signer must not be revoked at period j: for all $B_{ij} = h_j^{s_i} \in RL_j$,

$$T_2 \neq e(B_{ij}, T_1) \tag{F.16}$$

As in all VLR-GS schemes, there is an implicit tracing algorithm that can determine which group member created a valid signature using the vector of revocation tokens (and the revocation test (F.16)) which acts as a tracing key. We observe that, if necessary, the group manager is able to explicitly open the signature in $O(1)$ time by performing a BBS-decryption of $\vec{\sigma_2}$ using the trapdoor information α_1, α_2.

As far as efficiency goes, signatures consist of 46 elements of $\mathbb{G}$ and 1 element of $\mathbb{G}_T$. If we consider an implementation using symmetric pairings with a 256-bit group order and also assume that elements of $\mathbb{G}_T$ have a 1024-bit representation (with symmetric pairings and supersingular curves, such pairing-values can even be compressed to the third of their length as suggested in [SB04]), we obtain signatures of about 1.56 kB.

F.3.2 Security

When proving the BU-anonymity property, it seems natural to use a sequence of games starting with the real attack game and ending with a game where T_2 is replaced by a random element of $\mathbb{G}_T$ so as to leave no advantage to the adversary while avoiding to affect the adversary's view provided the Decision Bilinear Diffie-Hellman (DBDH) assumption holds. The problem becomes to simulate (using a fake common reference string) the NIZK proof that (g, T_1, h_j, K_2, T_2) forms a bilinear Diffie-Hellman tuple. Since T_2 is a given element of $\mathbb{G}_T$ in the proof, there is apparently no way to simulate the proof for relation (F.12).

As a natural workaround to this problem, we use the Decision Tripartite Diffie-Hellman assumption instead of the DBDH assumption in the last transition of the sequence of games.

Theorem F.3.1 [BU-anonymity] The scheme satisfies the backward unlinkable anonymity assuming that the Decision Linear problem and the Decision Tripartite Diffie-Hellman problem are both hard in $\mathbb{G}$. More precisely, we have

$$\mathbf{Adv}_{\mathcal{A}}^{\text{bu-anon}}(\lambda) \leq T \cdot N \cdot \left(2 \cdot \mathbf{Adv}_{\mathbb{G}}^{\text{DLIN}}(\lambda) + \mathbf{Adv}_{\mathbb{G}}^{\text{DTDH}}(\lambda)\right) \tag{F.17}$$

where N is the maximal number of users and T is the number of time periods.

Proof: The proof is a sequence of games organized in such a way that even an unbounded adversary has no advantage in the final game while the first one is the real attack game as captured by definition F.2.4. Throughout the sequence, we call S_i the event that the adversary wins and her advantage is $Adv_i = |\Pr[S_i] - 1/2|$.

Game 1: the challenger $\mathcal{B}$ sets up the scheme by choosing random exponents

$$\omega, \alpha, \alpha_1, \alpha_2, \xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$$

and setting g^ω and $A = e(g,g)^\alpha$. It also sets $u = g^\gamma$ for a randomly chosen $\gamma \xleftarrow{R} \mathbb{Z}_p^*$ and picks $h_1, \ldots, h_T \in \mathbb{G}$ as well as vectors $\overline{v} \in \mathbb{G}^{n+1}$, and defines $\vec{g_1} = (g_1 = g^{\alpha_1}, 1, g)$, $\vec{g_2} = (1, g_2 = g^{\alpha_2}, g)$, $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$. Using ω, α, it generates users' private keys and answers all queries as in the real game. At the challenge phase, the adversary chooses two unrevoked and uncorrupted users $i_0^\star, i_1^\star$ and is given a challenge signature $\sigma^\star$ on behalf of signer $i_{d^\star}^\star$. Eventually, she outputs a guess $d' \in \{0,1\}$ and her advantage is $Adv_1 = |\Pr[S_1] - 1/2|$, where S_1 denotes the event that $d' = d^\star$.

Game 2: we modify the simulation and let the simulator $\mathcal{B}$ pick two indices $i^\star \in \{1, \ldots, N\}, j^\star \xleftarrow{R} \{1, \ldots, T\}$ at the outset of the simulation. In the challenge phase, $\mathcal{B}$ aborts if $\mathcal{A}$'s chosen pair $(i_0^\star, i_1^\star)$ does not contain $i^\star$ or if $\mathcal{A}$ does not choose to be challenged for period $j^\star$. It also fails if $i^\star$ is ever queried for corruption or if it is queried for revocation before or during period $j^\star$. Assuming that $\mathcal{B}$ is lucky when drawing $i^\star, j^\star$ (which is the case with probability $(2/N) \cdot (1/T)$ since $i^\star$ and $j^\star$ are independent of $\mathcal{A}$'s view), the introduced failure event does not occur. We can write $Adv_2 = 2 \cdot Adv_1/(NT)$.

Game 3: we introduce a new rule that causes $\mathcal{B}$ to abort. At the challenge step, we have $i^\star \in \{i_0^\star, i_1^\star\}$ unless the failure event of Game 2 occurs. The new rule is the following: when $\mathcal{B}$ flips $d^\star \xleftarrow{R} \{0,1\}$, it aborts if $i_{d^\star}^\star \neq i^\star$. With probability 1/2, this rule does not apply and we have $Adv_3 = 1/2 \cdot Adv_2$.

Game 4: we modify the setup phase and consider group elements $Z_1 = g^{z_1}$, $Z_2 = g^{z_2}$ that are used to generate the public key gpk and users' private keys. Namely, for $j \in \{1, \ldots, T\} \backslash \{j^\star\}$, $\mathcal{B}$

chooses $\mu_j \xleftarrow{R} \mathbb{Z}_p^*$ and defines $h_j = g^{\mu_j}$ whereas it sets $h_{j^\star} = Z_2$. Also, $\mathcal{B}$ chooses $\nu \xleftarrow{R} \mathbb{Z}_p^*$ and sets $A = e(g, Z_1 \cdot g^{\omega})^{\nu}$ (so that α is implicitly fixed as $\alpha = \nu(z_1 + \omega)$). Private keys of users $i \neq i^\star$ are calculated as $(K_1, K_2, K_3) = \big((Z_1 \cdot g^{\omega})^{\nu/(\omega+s_i)}, g^{s_i}, u^{s_i}\big)$, for a random $s_i \xleftarrow{R} \mathbb{Z}_p^*$ and using ω. Since $\mathcal{B}$ knows s_i for each $i \neq i^\star$, it can compute revocation tokens $B_{ij} = h_j^{s_i}$ for users $i \neq i^\star$ in any period.

The group signing key of the expected target user $i^\star$ is set as the triple

$$(K_1, K_2, K_3) = (g^{\nu}, Z_1, Z_1^{\gamma}),$$

which implicitly defines $s_{i^\star} = z_1 = \log_g(Z_1)$. We note that, for periods $j \neq j^\star$, the revocation tokens $h_j^{s_{i^\star}}$ are also computable as $Z_2^{\mu_j}$. On the other hand, the token $h_{j^\star}^{s_{i^\star}} = g^{z_1 z_2}$ is not computable from Z_1, Z_2. However, unless the abortion rule of Game 2 occurs, $\mathcal{A}$ does not query it. Although $\mathcal{B}$ does not explicitly use $z_1 = \log_g(Z_1)$ and $z_2 = \log_g(Z_2)$, it still knows all users' private keys and it can use them to answer signing queries according to the specification of the signing algorithm. It comes that $\mathcal{A}$'s view is not altered by these changes and we have $\Pr[S_4] = \Pr[S_3]$.

Game 5: we bring a new change to the setup phase and generate the CRS $(\vec{g_1}, \vec{g_2}, \vec{g_3})$ by setting $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2} \odot \iota(g)^{-1}$ instead of $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$. We note that vectors $\vec{g_1}, \vec{g_3}, \vec{g_3}$ are now linearly independent. Any noticeable change in the adversary's behavior is easily seen[1] to imply a statistical test for the Decision Linear problem so that we can write $|\Pr[S_5] - \Pr[S_4]| = 2 \cdot \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$.

Game 6: we modify the generation of the challenge signature and use the trapdoor (ξ_1, ξ_2) of the CRS to simulate NIZK proofs. We suppose that $\mathcal{B}$ knows values $(Z_1, Z_2, Z_3) = (g^{z_1}, g^{z_2}, g^{z_3})$ and $\eta = g^{z_1 z_2 z_3}$. Elements Z_1 and Z_2 are used to define the group public key as in Game 4 whereas Z_3 will be used to create the challenge signature on behalf of user $i^\star$ for period $j^\star$. To this end, $\mathcal{B}$ first implicitly defines $\delta = z_3$ by setting

$$T_1 = Z_3 \qquad\qquad T_2 = e(g, \eta).$$

Elements $\theta_1, \ldots, \theta_4$ are committed to as specified by the scheme and π_1, π_2 are calculated accordingly. This time however, $\vec{\sigma_5}$ is calculated as a commitment to $1_{\mathbb{G}}$: namely, $\vec{\sigma_5} = \vec{g_1}^{r_5} \odot \vec{g_2}^{s_5} \odot \vec{g_3}^{t_5}$, where $r_5, s_5, t_5 \xleftarrow{R} \mathbb{Z}_p^*$. Then, $\mathcal{B}$ generates a proof $\pi_3 = (\vec{\pi}_{3,1}, \vec{\pi}_{3,2}, \vec{\pi}_{3,3})$ satisfying

$$F(\vec{\sigma_2}, \vec{\sigma_5}) = F(\iota(g), \iota(\eta)) \odot F(\vec{\pi}_{3,1}, \vec{g_1}) \odot F(\vec{\pi}_{3,2}, \vec{g_2}) \odot F(\vec{\pi}_{3,3}, \vec{g_3}). \tag{F.18}$$

Such an assignment can be obtained as

$$\vec{\pi}_{3,1} = \vec{\sigma_2}^{r_5} \odot \iota(\eta)^{-\xi_1} \qquad \vec{\pi}_{3,2} = \vec{\sigma_2}^{s_5} \odot \iota(\eta)^{-\xi_2} \qquad \vec{\pi}_{3,3} = \iota(\eta) \odot \vec{\sigma_2}^{t_5}.$$

We note that the value $\theta_5 = h_{j^\star}^{\delta} = g^{z_2 z_3}$ is not used by $\mathcal{B}$. To simulate the proof π_3 that $T_2 = e(\theta_2, \theta_5)$ without knowing θ_5, the simulator takes advantage of the fact that $T_2 = e(g, \eta)$ for known $g, \eta \in \mathbb{G}$ (and simulating such a proof would not have been possible if T_2 had been a given element of $\mathbb{G}_T$). To simulate proofs $\pi_4 = (\pi_{4,1}, \pi_{4,2}, \pi_{4,3})$, $\pi_5 = (\pi_{5,1}, \pi_{5,2}, \pi_{5,3})$ that relations (F.14)-(F.15) are both satisfied, $\mathcal{B}$ generates π_4 as if it were a real proof using the variable assignment $\theta_5 = \theta_6 = 1_{\mathbb{G}}$ that obviously satisfies $e(\theta_6, T_1) = e(g, \theta_5)$ (and $\vec{\sigma_6} = \vec{g_1}^{r_6} \odot \vec{g_2}^{s_6} \odot \vec{g_3}^{t_6}$ is thus computed as a commitment to $1_{\mathbb{G}}$). As for π_5, the assignment

$$\pi_{5,1} = g^{r_6} \cdot h_j^{-\xi_1} \qquad \pi_{5,2} = g^{s_6} \cdot h_j^{-\xi_2} \qquad \pi_{5,3} = g^{t_6} \cdot h_j.$$

[1] Indeed, $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^{\xi_1+\xi_2}) = 1]$ and $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^{\xi_1+\xi_2-1}) = 1]$ are both within distance $\mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$ from $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^{z}) = 1]$, where z is random.

is easily seen to satisfy the last verification equation

$$E(g, \vec{\sigma_6}) = E(h_j, \iota(g)) \odot E(\pi_{5,1}, \vec{g_1}) \odot E(\pi_{5,2}, \vec{g_2}) \odot E(\pi_{5,3}, \vec{g_3})$$

since $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2} \odot \iota(g)^{-1}$. Simulated proofs π_4, π_5 are then randomized as explained in [GS08] to be uniform in the space of valid proofs and achieve perfect witness indistinguishability. Simulated proofs are perfectly indistinguishable from real proofs and $\Pr[S_6] = \Pr[S_5]$.

Game 7: is identical to Game 6 but we replace η (that was equal to $g^{z_1 z_2 z_3}$ in Game 6) by a random group element. It is clear that, under the DTDH assumption, this change does not significantly alter $\mathcal{A}$'s view. We thus have $|\Pr[S_7] - \Pr[S_6]| \leq \mathbf{Adv}^{\mathrm{DTDH}}_{\mathbb{G},\mathcal{B}}(\lambda)$.
In Game 7, it is easy to see that $\Pr[S_7] = 1/2$. Elements T_1 and T_2 are indeed completely independent of $s_{i^\star} = z_1$ (and thus of $i^\star$). Moreover, in the WI setting, all commitments $\vec{\sigma_1}, \ldots, \vec{\sigma_5}$ are perfectly hiding and proofs $\pi_1, \ldots, \pi_5$ reveal no information on underlying witnesses.
When gathering probabilities, we obtain the upper bound (F.17) on $\mathcal{A}$'s advantage in Game 1. ∎

Theorem F.3.2 [Traceability] The scheme satisfies the full non-traceability assuming that the N-Hidden Strong Diffie-Hellman problem is hard in $\mathbb{G}$. More precisely, we have

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathcal{A}}(\lambda) \leq 4 \cdot n \cdot N \cdot q_s \cdot \left(1 - \frac{(N-1)}{p}\right)^{-1} \cdot \left(\mathbf{Adv}^{N\text{-HSDH}}(\lambda) + \mathbf{Adv}^{\mathrm{CR}}(n)\right) \quad \text{(F.19)}$$

where N is maximum of the number of the adversary signature queries and the maximal number of users and T is the number of time periods.

Proof: The proof is very similar to the proof of full traceability in the Boyen-Waters [BW07] group signature. One difference is that [BW07] reduces the full traceability property of their scheme to the unforgeability of a 2-level hierarchical signature [KMPR05]. To prove this result, Boyen and Waters restricted the message space (where the element s_i, that uniquely identifies the group member is the group signature, must be chosen) to a relatively small interval at the first level.
In our proof of anonymity, we need elements s_i to be uniformly chosen in $\mathbb{Z}_p^*$. Therefore, we cannot directly link the security of our scheme to that of the 2-level hierarchical signature of [BW07] and a direct proof is needed (but it is simply obtained using the techniques from [BW07]). Namely, two kinds of forgeries must be considered as in [BW07]:

- **Type I forgeries** are those for which the implicit tracing algorithm fails to identify the signer using the vector of revocation tokens for the relevant period $j^\star$.
- **Type II forgeries** are those for which the implicit tracing algorithm incriminates a user outside the coalition and that was not requested to sign the message $M^\star$ during period $j^\star$.

The two kinds of adversaries are handled separately in lemmas F.3.3 and F.3.4.
To conclude the proof, we consider an algorithm $\mathcal{B}$ that guesses the kind of forgery that $\mathcal{A}$ will come up with. Then, $\mathcal{B}$ runs the appropriate HSDH solver among those described in previous lemmas. If the guess is correct, $\mathcal{B}$ solves the HSDH problem with the success probability given in the lemmas. Since this guess is correct with probability 1/2, we obtain the claimed security bound. ∎

Lemma F.3.3 If N is the maximal number of users, any Type I forger $\mathcal{A}$ has no advantage than $\mathbf{Adv}_{\mathcal{A}}^{\text{Type-I}}(\lambda) \leq \mathbf{Adv}^{N\text{-HSDH}}(\lambda)$.

Proof: The proof is close to the one of lemma A.1 in [BW07]. The simulator $\mathcal{B}$ is given a N-HSDH instance consisting of elements $(g, \Omega = g^{\omega}, u)$ and triples

$$\{(A_i, B_i, C_i) = (g^{1/(\omega+s_i)}, g^{s_i}, u^{s_i})\}_{i=1,\ldots,N}.$$

The simulator picks $\alpha, \beta_0, \ldots, \beta_n \xleftarrow{R} \mathbb{Z}_p^*$ and sets $v_i = g^{\beta_i}$, for $i = 0, \ldots, n$. Vectors $\vec{g_1}, \vec{g_2}, \vec{g_3}$ are chosen as $\vec{g_1} = (g_1 = g^{\alpha_1}, 1, g)$, $\vec{g_2} = (1, g_2 = g^{\alpha_2}, g)$ and $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$, for randomly chosen $\alpha_1, \alpha_2, \xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$, in such a way that the CRS $\mathbf{g} = (\vec{g_1}, \vec{g_2}, \vec{g_3})$ provides perfectly sound proofs for which $\mathcal{B}$ retains the extraction trapdoor $(\alpha_1 = \log_g(g_1), \alpha_2 = \log_g(g_2))$. Finally, $\mathcal{B}$ generates $(h_1, \ldots, h_T) \in \mathbb{G}^T$ as $h_j = g^{\zeta_j}$, for $j = 1, \ldots, T$, with $\zeta_1, \ldots, \zeta_T \xleftarrow{R} \mathbb{Z}_p^*$. Then, $\mathcal{B}$ starts interacting with the Type I adversary $\mathcal{A}$ who is given the group public key $\mathsf{gpk} := (g, A = e(g,g)^{\alpha}, h_1, \ldots, h_T, \Omega, u, \overline{v}, \mathbf{g})$ and the vector of revocation tokens grt, which $\mathcal{B}$ generates as $\mathsf{grt}[i][j] = h_j^{s_i} = B_i^{\zeta_j}$. The simulation proceeds as follows:

- when $\mathcal{A}$ decides to corrupt user $i \in \{1, \ldots, N\}$, $\mathcal{B}$ returns the HDSH triple (A_i, B_i, C_i).
- when $\mathcal{A}$ queries a signature from user $i \in \{1, \ldots, N\}$ for a message M, $\mathcal{B}$ uses the private key $(K_1, K_2, K_3) = (A_i, B_i, C_i)$, to generate the signature by following the specification of the signing algorithm.

When $\mathcal{A}$ outputs her forgery $(M^{\star}, j^{\star}, \sigma^{\star})$, $\mathcal{B}$ uses elements α_1, α_2 to decrypt $\vec{\sigma_i}^{\star}$, for indices $i \in \{1, \ldots, 5\}$, and obtain $\theta_1^{\star} = (g^{\alpha})^{1/(\omega+s^{\star})}$, $\theta_2^{\star} = g^{s^{\star}}$ as well as $\theta_3^{\star} = u^{s^{\star}} \cdot (v_0 \cdot \prod_{k=1}^{n} v_k^{m_k})^r$ and $\theta_4^{\star} = g^r$. From these values, $\mathcal{B}$ can extract $u^{s^{\star}}$ since it knows the discrete logarithm $\log_g(v_0 \cdot \prod_{k=1}^{n} v_k^{m_k}) = \beta_0 + \sum_{k=1}^{n} m_k \beta_k$, where $m_1 \ldots m_n = H(j^{\star} || M^{\star}) \in \{0,1\}^n$. Since $\sigma^{\star}$ is a Type I forgery, the implicit tracing algorithm must fail to identify one of the group members $\{1, \ldots, N\}$. The perfect soundness of the proof system implies that $s^{\star} \notin \{s_1, \ldots, s_N\}$ and $(\theta_1^{\star 1/\alpha}, \theta_2^{\star}, u^{s^{\star}})$ is necessarily an acceptable solution. ∎

Lemma F.3.4 The scheme is secure against Type II forgeries under the $(N-1)$-HSDH assumption. The advantage of any Type II adversary $\mathcal{A}$ is at most

$$\mathbf{Adv}_{\mathcal{A}}^{\text{Type-II}}(\lambda, n) \leq 2 \cdot n \cdot N \cdot q_s \cdot \Big(1 - \frac{(N-1)}{p}\Big)^{-1} \cdot \Big(\mathbf{Adv}^{(N-1)\text{-HSDH}}(\lambda) + \mathbf{Adv}^{\text{CR}}(n)\Big)$$

where N and q_s stand for the number of users and the number of signing queries, respectively, and the last term accounts for the probability of breaking the collision-resistance of H.

Proof: The proof is based on lemma A.2 in [BW07]. Namely, the simulator $\mathcal{B}$ receives a $(N-1)$-HSDH input comprising $(g, \Omega = g^{\omega}, u)$ and a set of triples

$$\{(A_i, B_i, C_i) = (g^{1/(\omega+s_i)}, g^{s_i}, u^{s_i})\}_{i=1,\ldots,N-1}.$$

To prepare the public key gpk, the simulator $\mathcal{B}$ picks a random index $\nu \xleftarrow{R} \{0, \ldots, n\}$, as well as $\rho_0, \ldots, \rho_n \xleftarrow{R} \mathbb{Z}_p^*$ and integers $\beta_0, \ldots, \beta_n \xleftarrow{R} \{0, \ldots, 2q_s - 1\}$. It sets $v_0 = u^{\beta_0 - 2\nu q_s} \cdot g^{\rho_0}$, $v_i = u^{\beta_i} \cdot g^{\rho_i}$ for $i = 1, \ldots, n$. It also defines $h_1, \ldots, h_T$ by setting $h_j = g^{\zeta_j}$, with $\zeta_j \xleftarrow{R} \mathbb{Z}_p^*$, for $j = 1, \ldots, T$. It finally chooses vectors $\mathbf{g}$ as specified by the setup algorithm to obtain perfectly sound proofs. Before starting its interaction with the Type II forger $\mathcal{A}$, $\mathcal{B}$ initializes a counters $ctr \leftarrow 0$ and

chooses an index $i^\star \xleftarrow{R} \{1,\ldots,N\}$ as a guess for the honest user on behalf of which $\mathcal{A}$ will attempt to generate a forgery. The simulation proceeds by handling $\mathcal{A}$'s queries in the following way.

Queries: at the first time that user $i \in \{1,\ldots,N\}$ is involved in a signing query or a corruption query, $\mathcal{B}$ does the following:

- if the query is a corruption query, $\mathcal{B}$ halts and declares failure if $i = i^\star$ as it necessarily guessed the wrong user $i^\star$. Otherwise, it increments ctr and returns the triple $(A_{ctr}, B_{ctr}, C_{ctr})$ as a private key for user (K_1, K_2, K_3).

- if the query is a signing query for period $j \in \{1,\ldots,T\}$,

  - if $i \neq i^\star$ $\mathcal{B}$ increments ctr and answers the query by running the signing algorithm using the private key $(K_1, K_2, K_3) = (A_{ctr}, B_{ctr}, C_{ctr})$.
  - if $i = i^\star$, $\mathcal{B}$ chooses $t^\star \xleftarrow{R} \mathbb{Z}_p^*$ at random and implicitly defines a triple $(K_1^\star, K_2^\star, K_3^\star) = (g^{1/t^\star}, g^{t^\star} \cdot \Omega^{-1}, *)$, where $*$ is a placeholder for an unknown group element (note that this implicitly defines $s^* = t^\star - \omega$). Then, $\mathcal{B}$ computes $m_1 \ldots m_n = H(j||M) \in \{0,1\}^n$. At this stage, it is convenient to write $F(m_1 \ldots m_n) = v_0 \cdot \prod_{k=1}^n v_k^{m_k}$ as $F(m_1 \ldots m_n) = u^J \cdot g^K$ where $J = \beta_0 - 2\nu q_s + \sum_{j=1}^n \beta_j m_j$, $K = \rho_0 + \sum_{j=1}^n \rho_j m_j$. If $J = 0$, $\mathcal{B}$ aborts. Otherwise, it can pick $r \xleftarrow{R} \mathbb{Z}_p^*$ and compute a pair

$$\Big(\theta_3 = u^{t^\star} \cdot F(m_1 \ldots m_n)^r \cdot \Omega^{\frac{K}{J}},\ \theta_4 = g^r \cdot \Omega^{\frac{1}{J}}\Big),$$

    which can be re-written as $(\theta_4 = u^{t^\star - \omega} \cdot F(m_1 \ldots m_n)^{\tilde{r}}, \theta_5 = g^{\tilde{r}})$ if we define $\tilde{r} = r + \omega/J(\mathsf{m})$. This pair then allows generating a suitably anonymized signature. In particular, since $\mathcal{B}$ knows $\theta_2 = K_2^\star = g^{t^\star} \cdot \Omega^{-1}$, it is able to compute $T_2 = e(h_j, K_2^\star)^\delta$ and $T_1 = g^\delta$ for a random $\delta \xleftarrow{R} \mathbb{Z}_p^*$.

When subsequent queries involve the same user i, $\mathcal{B}$ responds as follows (we assume that corruption queries are distinct):

- For corruption queries on users $i \in \{1,\ldots,N\}$ that were previously involved in signing queries, $\mathcal{B}$ aborts if $i = i^\star$. Otherwise, it knows the private key (K_1, K_2, K_3) (that was used to answer signing queries) and hands it to $\mathcal{A}$.

- For signing queries, $\mathcal{B}$ uses the same values as in the first query involving the user $i \in \{1,\ldots,N\}$. If $i \neq i^\star$, $\mathcal{B}$ uses the same triple $(A_{ctr}, B_{ctr}, C_{ctr})$. In the case $i = i^\star$, $\mathcal{B}$ re-uses the pair $(K_1^\star, K_2^\star) = (g^{1/t^\star}, g^{t^\star} \cdot \Omega^{-1})$ and proceeds as in the first query involving $i^\star$ (but uses a fresh random exponent r).

Forgery: the game ends with the adversary outputting message $M^\star$ together with a type II forgery $\sigma^\star = (T_1^\star, T_2^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_6}^\star, \pi_1^\star, \ldots, \pi_5^\star)$ for some period $j^\star \in \{1,\ldots,T\}$. By assumption, the implicit tracing algorithm must point to some user who did not sign $M^\star$ at period $j^\star$. Then, $\mathcal{B}$ halts and declares failure if $\sigma^\star$ does not trace to user $i^\star$. Since the chosen index $i^\star$ was independent of $\mathcal{A}$'s view, with probability $1/N$, $\mathcal{B}$'s guess turns out to be correct. Then, the perfect soundness of the proof system implies that $\vec{\sigma_2}^\star$ is a BBS encryption of $K_2^\star$. Then, $\mathcal{B}$ computes $\mathsf{m}^\star = m_1 \ldots m_n = H(j^\star||M^\star)$. If user $i^\star$ signed a message M at period j such that

$(j, M) \neq (j^\star, M^\star)$ but $H(j||M) = H(j^\star||M^\star)$, $\mathcal{A}$ was necessarily able to generate a collision on H. Otherwise, the perfect soundness of the proof system implies that $\vec{\sigma_3}^\star$ and $\vec{\sigma_4}^\star$ decrypt into

$$\theta_3^\star = u^{t^\star - \omega} F(\mathsf{m}^\star)^r \qquad\qquad \theta_4^\star = g^r$$

for some $r \in \mathbb{Z}_p^*$ and where $F(\mathsf{m}^\star) = v_0 \cdot \prod_{k=1}^n v_k^{m_k} = u^{J^\star} \cdot g^{K^\star}$ and $s_\star = t_{i^\star} - \omega$. Then, $\mathcal{B}$ aborts if $J(\mathsf{m}^\star) = \beta_0 + \sum_{j=1}^n \beta_j m_j - 2\nu q_s \neq 0$. Otherwise, $\mathcal{B}$ can compute u^{s^*} and thereby obtains a full tuple $(g^{1/(\omega+s^\star)},\ g^{s^\star},\ u^{s^\star})$ where $s^\star = t^\star - \omega$ differs from $s_1, \ldots, s_{N-1}$ with probability at least $1 - (N-1)/p$ (since the value $t^\star$ was chosen at random).
$\mathcal{B}$'s probability not to abort throughout the simulation can assessed as in [Wat05, BW07]. More precisely, one can show that $J \neq 0$ in all signing queries with probability greater than $1/2$. Conditionally on the event that $\mathcal{B}$ does not abort before the forgery stage, the probability to have $J^\star = 0$ is then shown to be at least $1/(2nq_s)$ (see [Wat05, BW07] for details). ▮

F.3.3 A Variant with Shorter Group Public Keys

As described in this section, the scheme suffers from a group public key of size $O(T)$, which makes it impractical when the number of time periods is very large. In the random oracle model $h_1, \ldots, h_T$ could be derived from a random oracle. However, avoiding the dependency on T in the group public key size is also possible without resorting to random oracles. This can be achieved using the techniques introduced in [BB04a] in the context of identity-based encryption.

The vector $(h_1, \ldots, h_T)$ is replaced by a triple $(h, h_0, h_1) \in \mathbb{G}^3$ and the revocation token of user i at period $j \in \{1, \ldots, T\}$ is defined to be the pair $(B_{ij1}, B_{ij2}) = (h^{s_i} \cdot F(j)^\rho, g^\rho)$, where $\rho \xleftarrow{R} \mathbb{Z}_p^*$ and $F(j) = h_0 \cdot h_1^j$ is the selectively-secure identity-hashing function of Boneh and Boyen [BB04a]. Since the revocation token (B_{ij1}, B_{ij2}) satisfies the relation $e(B_{ij1}, g) = e(h, g^{s_i}) \cdot e(F(j), B_{ij2})$, we have $e(B_{ij1}, g^\delta) = e(h, g^{s_i})^\delta \cdot e(F(j)^\delta, B_{ij2})$ for any $\delta \in \mathbb{Z}_p^*$.

Therefore, in each signature σ, the pair (T_1, T_2) is superseded by a triple $(T_1, T_2, T_3) = (g^\delta, F(j)^\delta, e(h, K_2)^\delta)$ (so that the verifier needs the check that $e(T_1, F(j)) = e(g, T_2)$) whereas $\vec{\sigma_5}$ becomes a commitment to $\theta_5 = h^\delta$ and the NIZK proof for relation (F.13) is replaced by a proof that $e(h, T_1) = e(g, \theta_5)$. At step 2 of the verification algorithm, the revocation test then consists in testing whether $e(T_1, B_{ij1}) = T_3 \cdot e(T_2, B_{ij2})$ for revocation tokens $\{(B_{ij1}, B_{ij2})\}_{i \in RL_j}$. Using the technique of [BB04a] to generate tokens for periods $j \in \{1, \ldots, T\} \backslash \{j^\star\}$, it can be checked that everything goes through in the proof of anonymity.

F.4 Conclusion

We described a simple way to provide Boyen-Waters group signatures with an efficient verifier local revocation mechanism with backward unlinkability.

The scheme can be easily extended so as to provide exculpability (and prevent the group manager from signing on behalf of users) using a dynamic joining protocol such as the one of [LY09]. It would be interesting to turn the scheme into a traceable signature [KTY04] supporting fine-grained (*i.e.* per period) user tracing while leaving users the ability to claim their signatures.

Smooth Projective Hash Proof Systems and Applications

Appendix G:

Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions

TCC 12

Olivier Blazy, David Pointcheval and Damien Vergnaud

This article demonstrates that the notion of smooth projective hash functions can be useful to design round-optimal privacy-preserving interactive protocols (oblivious signature-based envelopes and blind signatures). It shows that this approach is suitable for designing schemes that rely on standard security assumptions in the standard model with a common-reference string and are more efficient than those obtained using the Groth-Sahai methodology.

Appendix H:

Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages

PKC 2013

Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval and Damien Vergnaud

This article provides a general framework (in the Universal Composability *setting), that encompasses several previous Authenticated Key-Exchange primitives such as (Verifier-based) Password-Authenticated Key Exchange or Secret Handshakes, we call* LAKE *for Language-Authenticated Key Exchange. It presents smooth projective hash functions on new languages, whose efficient implementations are of independent interest and very practical realizations of Secret Handshakes and Credential-Authenticated Key Exchange protocols.*

Appendix I:

New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange

Crypto 2013

Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval and Damien Vergnaud

This article proposes a new efficient smooth projective hash function on Cramer-Shoup ciphertexts that leads to the design of the most efficient Password-Authenticated Key Exchange (PAKE) known so far: a one-round PAKE with two simultaneous flows consisting of 6 group elements each only, in any DDH*-group without any pairing. It also presents a generic construction for smooth projective hash functions, in order to check the validity of complex relations on encrypted values. This allows to extend this work on PAKE to the more general family of Langage-Authenticated Key Exchange but also to blind signatures.*

Appendix G

Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions

TCC 2012
[BPV12b] with O. Blazy and D. Pointcheval

Abstract : In 2008, Groth and Sahai proposed a powerful suite of techniques for constructing non-interactive zero-knowledge proofs in bilinear groups. Their proof systems have found numerous applications, including group signature schemes, anonymous voting, and anonymous credentials. In this paper, we demonstrate that the notion of *smooth projective hash functions* can be useful to design round-optimal privacy-preserving interactive protocols. We show that this approach is suitable for designing schemes that rely on standard security assumptions in the standard model with a common-reference string and are more efficient than those obtained using the Groth-Sahai methodology. As an illustration of our design principle, we construct an efficient oblivious signature-based envelope scheme and a blind signature scheme, both round-optimal.

G.1 Introduction

In 2008, Groth and Sahai [GS08, GS12] proposed a way to produce efficient and practical non-interactive zero-knowledge and non-interactive witness-indistinguishable proofs for (algebraic) statements related to groups equipped with a bilinear map. They have been significantly studied in cryptography and used in a wide variety of applications in recent years (*e.g.* group signature schemes [BW06b,BW07,Gro07] or blind signatures [AFG$^+$10,BFPV11]). While avoiding expensive NP-reductions, these proof systems still lack in practicality and it is desirable to provide more efficient tools.

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. If it is hard to distinguish elements of the special subset from non-elements, then this primitive can

be seen as special type of zero-knowledge proof system for membership in the special subset. The notion of SPHF has found applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09]). We present some other applications with privacy-preserving primitives that were already inherently interactive.

Applications: Our two applications are *Oblivious Signature-Based Envelope* [LDB03] and *Blind Signatures* [Cha82].

Oblivious Signature-Based Envelope (OSBE) were introduced in [LDB03]. It can be viewed as a nice way to ease the asymmetrical aspect of several authentication protocols. Alice is a member of an organization and possesses a certificate produced by an authority attesting she is in this organization. Bob wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob sends an obfuscated version of this message P to Alice, in such a way that Alice will be able to find P if and only if Alice is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization. They are part of a growing field of protocols, around *automated trust negotiation*, which also include Secret Handshakes [BDS+03], Password-based Authenticated Key-Exchange [GL06], and Hidden Credentials [BHS04]. Those schemes are all closely related, so due to space constraints, we are going to focus on OSBE (as if you tweak two of them, you can produce any of the other protocols [CJT04]).

Blind signatures were introduced by Chaum [Cha82] for electronic cash in order to prevent the bank from linking a coin to its spender: they allow a user to obtain a signature on a message such that the signer cannot relate the resulting message/signature pair to the execution of the signing protocol. In [Fis06], Fischlin gave a generic construction of round-optimal blind signatures in the common-reference string (CRS) model: the signing protocol consists of one message from the user to the signer and one response by the signer. The first practical instantiation of round-optimal blind signatures in the standard model was proposed in [AFG+10] but it relies on non-standard computational assumptions. We proposed, recently only [BFPV11], the most efficient realizations of round-optimal blind signatures in the common-reference string model under classical assumptions. But these schemes still use the Groth-Sahai proof systems.

Contributions: Our first contribution is to clarify and increase the security requirements of an OSBE scheme. The main improvement residing in some protection for both the sender and the receiver against the Certification Authority. The OSBE notion echoes directly to the idea of SPHF if we consider the language $\mathcal{L}$ defined by encryption of valid signatures, which is hard to distinguish under the security of the encryption schemes. We show how to build, from a SPHF on this language, an OSBE scheme in the standard model with a CRS. And we prove the security of our construction in regards of the security of the commitment (the ciphertext), the signature and the SPHF scheme. We then show how to build a simple and efficient OSBE scheme relying on a classical assumption, DLin. An asymmetrical version is also sketched in the Appendix G.7.2: the communication cost is divided by two. To build those schemes, we use SPHF in a new way, avoiding the need of costly Groth-Sahai proofs when an interaction is inherently needed in the primitive. Our method does not add any other interaction, and so supplement smoothly those proofs.

To show the efficiency of the method, and the ease of application, we then adapt two Blind Signature schemes proposed in [BFPV11]. Our approach fits perfectly and decreases significantly the communicational complexity of the schemes (it is divided by more than three in one construction). Moreover one scheme relies on a weakened security assumptions: the XDH assumption instead of the SXDH assumption and permits to use more bilinear group settings

(namely, Type-II and Type-III bilinear groups [GPS08] instead of only Type-III bilinear groups for the construction presented in [BFPV11]).

Organization. The paper is divided into three main parts after a brief recall of standard definitions and security notions. In a first part, we present a high-level version of our OSBE protocol, and prove its security. We then instantiate this protocol with Linear encryption, Waters signature and study its efficiency when compared with existing versions. In a last part, we continue to use SPHF as an effective replacement to proofs of knowledge to instantiate a blind signature. In the appendices, we provide details on our instantiation of SPHF, the detailed security proofs, and a sketch of the asymmetric instantiations of our OSBE scheme and the blind signature.

G.2 Definitions

In this section, we briefly recall the notations and the security notions of the basic primitives we will use in the rest of the paper, and namely public key encryption, signature and smooth projective hash functions (SPHF), using the Gennaro-Lindell [GL03] extension. More formal definitions are provided in the Appendix G.5.1, together with concrete instantiations (linear encryption, Waters signature, SPHF on linear tuples) and the computational assumptions in the Appendix G.5.3. In a second part, we recall and enhance the security model of oblivious signature-based envelope protocols [LDB03].

G.2.1 Notations

Encryption Scheme. An encryption scheme $\mathcal{E}$ is defined by four algorithms: $\mathsf{ESetup}(1^k)$ that generates the global parameters param, $\mathsf{EKeyGen}(\mathsf{param})$ that generates the pair of encryption/decryption keys $(\mathsf{ek}, \mathsf{dk})$, $\mathsf{Encrypt}(\mathsf{ek}, m; r)$ that produces a ciphertext c, and $\mathsf{Decrypt}(\mathsf{dk}, c)$ that decrypts it back. The security of an encryption scheme is defined through the semantic security (indistinguishability of ciphertexts against chosen-plaintext attacks) [GM84, BDPR98]: after having chosen two messages M_0, M_1 and received the encryption c of one of them, the adversary should be unable to guess which message has been encrypted. More precisely, we will use commitment schemes (as in [ACP09]), which should be hiding (indistinguishability) and binding (one opening only), with the additional extractability property. The latter property thus needs an extracting algorithm that corresponds to the decryption algorithm. Hence the notation with encryption schemes.

Signature Scheme. A signature scheme $\mathcal{S}$ is also defined by four algorithms: $\mathsf{SSetup}(1^k)$ that generates the global parameters param, $\mathsf{SKeyGen}(\mathsf{param})$ that generates a pair of verification/signing keys $(\mathsf{vk}, \mathsf{sk})$, $\mathsf{Sign}(\mathsf{sk}, m; s)$ that produces a signature σ, and $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ that checks its validity. The security of a signature scheme is defined by the unforgeability property (existential unforgeability against adaptive chosen-message attacks) [GMR88]. An adversary against the unforgeability tries to generate a valid signature on a message M of its choice, after a polynomial number of signing queries to the signer: the message M must be distinct from all the queries to the signing oracle.

Smooth Projective Hash Function. An SPHF system [CS02] on a language $\mathcal{L}$ is defined by five algorithms: $\mathsf{SPHFSetup}(1^k)$ that generates the global parameters, $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$ that generates a hashing key hk, $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ that derives the projection key hp, possibly depending on the word W [GL03, ACP09]. Then, $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ and

$\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$ outputs the hash value, either from the hashing key, or from the projection key and the witness. The correctness of the scheme assures that if W is indeed in $\mathcal{L}$ with w as a witness, then the two ways to compute the hash value give the same result. The security of a SPHF is defined through two different notions, the smoothness and the pseudo-randomness properties: The smoothness property guarantees that if $W \notin \mathcal{L}$, then the hash value is statistically random (statistically indistinguishable from a random element). The pseudo-randomness guarantees that even for a word $W \in \mathcal{L}$, but without the knowledge of a witness w, then the hash value is random (computationally indistinguishable from a random element). Abdalla *et al.* [ACP09] explained how to combine SPHF to deal with conjunctions and disjunctions of the languages. This is recalled in the Appendix G.5.2.

G.2.2 Oblivious Signature-Based Envelope

We now define an OSBE protocol, where a sender $\mathcal{S}$ wants to send a private message $P \in \{0,1\}^\ell$ to a recipient $\mathcal{R}$ in possession of a certificate/signature on a message M.

Definition G.2.1 [Oblivious Signature-Based Envelope] An Oblivious Signature-Based Envelope scheme is defined by four algorithms ($\mathsf{OSBESetup}, \mathsf{OSBEKeyGen}, \mathsf{OSBESign}, \mathsf{OSBEVerif}$), and one interactive protocol $\mathsf{OSBEProtocol}\langle\mathcal{S}, \mathcal{R}\rangle$:

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter, generates the global parameters param;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ generates the keys $(\mathsf{vk}, \mathsf{sk})$ of the certification authority;
- $\mathsf{OSBESign}(\mathsf{sk}, m)$ produces a signature σ on the input message m, under the signing key sk;
- $\mathsf{OSBEVerif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.
- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ between the sender $\mathcal{S}$ with the private message P, and the recipient $\mathcal{R}$ with a certificate σ. If σ is a valid signature under vk on the common message M, then $\mathcal{R}$ receives P, otherwise it receives nothing. In any case, $\mathcal{S}$ does not learn anything.

Such an OSBE scheme should be (the three last properties are additional —or stronger— security properties from the original definitions [LDB03]):

- *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
- *oblivious*: the sender should not be able to distinguish whether $\mathcal{R}$ uses a valid signature σ on M under vk as input. More precisely, if $\mathcal{R}_0$ knows and uses a valid signature σ and $\mathcal{R}_1$ does not use such a valid signature, the sender cannot distinguish an interaction with $\mathcal{R}_0$ from an interaction with $\mathcal{R}_1$;
- *(weakly) semantically secure*: the recipient learns nothing about $\mathcal{S}$ input P if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$;
- *semantically secure* (denoted sem): the above indistinguishability should hold even if the receiver has seen several interactions $\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ with valid signatures, and the same sender's input P;

$\mathsf{Exp}^{\mathsf{esc}-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Escrow Free property]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\texttt{INIT} : \mathsf{param})$
3. $(M, \sigma) \leftarrow \mathcal{A}(\texttt{FIND} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \mathcal{A}, \texttt{Rec}^*(\mathsf{vk}, M, \sigma, b)\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. `RETURN` b'

$\mathsf{Exp}^{\mathsf{sem}^*-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic security w.r.t. the authority]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\texttt{INIT} : \mathsf{param})$
3. $(M, \sigma, P_0, P_1) \leftarrow \mathcal{A}(\texttt{FIND} : \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{transcript} \leftarrow \mathsf{OSBEProtocol}\langle \texttt{Send}(\mathsf{vk}, M, P_b), \texttt{Rec}^*(\mathsf{vk}, M, \sigma, 0\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{transcript}, \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \texttt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. `RETURN` b'

$\mathsf{Exp}^{\mathsf{sem}-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic Security]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{OSBEKeyGen}(\mathsf{param})$
3. $(M, P_0, P_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{vk}, \texttt{Sign}^*(\mathsf{vk}, \cdot), \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}(\mathsf{vk}, \cdot, 0), \texttt{Exec}(\mathsf{vk}, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \texttt{Send}(\mathsf{vk}, M, P_b), \mathcal{A}\rangle$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \texttt{Sign}(\mathsf{vk}, \cdot), \texttt{Send}(\mathsf{vk}, \cdot, \cdot), \texttt{Rec}(\mathsf{vk}, \cdot, 0), \texttt{Exec}(\mathsf{vk}, \cdot, \cdot))$
6. `IF` $M \in \mathcal{SM}$ `RETURN` 0 `ELSE RETURN` b'

Figure G.1: Security Games for $\mathcal{OSBE}$

- *escrow free* (denoted esc): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input. This notion supersedes the above *oblivious* property, since this is basically oblivious w.r.t. the authority, without any restriction.

- *semantically secure w.r.t. the authority* (denoted sem^*): after the interaction, the authority (owner of the signing key sk) learns nothing about P.

We insist that the escrow-free property (esc) is stronger than the oblivious property, hence we will consider the former only. However, the semantic security w.r.t. the authority (sem^*) is independent from the basic semantic security (sem) since in the latter the adversary interacts with the sender whereas in the former the adversary (who generated the signing keys) has only passive access to a challenge transcript.

These security notions can be formalized by the security games presented on Figure G.1, where the adversary keeps some internal state between the various calls `INIT`, `FIND` and `GUESS`. They make use of the oracles described below, and the advantages of the adversary are, for all the security notions,

$$\mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{*-1}_{\mathcal{OSBE},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{*-0}_{\mathcal{OSBE},\mathcal{A}}(k) = 1]$$

$$\mathsf{Adv}^*_{\mathcal{OSBE}}(k, t) = \max_{\mathcal{A} \leq t} \mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k).$$

- $\texttt{Sign}(\mathsf{vk}, m)$: This oracle outputs a valid signature on m under the signing key sk associated to vk (where the pair $(\mathsf{vk}, \mathsf{sk})$ has been outputted by the $\mathsf{OSBEKeyGen}$ algorithm);

- $\texttt{Sign}^*(\mathsf{vk}, m)$: This oracle first queries $\texttt{Sign}(\mathsf{vk}, m)$. It additionally stores the query m to the list $\mathcal{SM}$;

- $\texttt{Send}(\mathsf{vk}, m, P)$: This oracle emulates the sender with private input P, and thus may consist of multiple interactions;
- $\texttt{Rec}(\mathsf{vk}, m, b)$: This oracle emulates the recipient either with a valid signature σ on m under the verification key vk (obtained from the signing oracle $\texttt{Sign}$) if $b = 0$ (as the above $\mathcal{R}_0$), or with a random string if $b = 1$ (as the above $\mathcal{R}_1$). This oracle is available when the signing key has been generated by $\mathsf{OSBEKeyGen}$ only;
- $\texttt{Rec}^*(\mathsf{vk}, m, \sigma, b)$: This oracle does as above, with a valid signature σ provided by the adversary. If $b = 0$, it emulates the recipient playing with σ; if $b = 1$, it emulates the recipient playing with a random string;
- $\texttt{Exec}(\mathsf{vk}, m, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ on m under the verification key vk (obtained from the signing oracle $\texttt{Sign}$). It basically activates the $\texttt{Send}(\mathsf{vk}, m, P)$ and $\texttt{Rec}(\mathsf{vk}, m, 0)$ oracles.
- $\texttt{Exec}^*(\mathsf{vk}, m, \sigma, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ (provided by the adversary). It basically activates the $\texttt{Send}(\mathsf{vk}, m, P)$ and $\texttt{Rec}^*(\mathsf{vk}, m, \sigma, 0)$ oracles.

Remark G.2.2 The OSBE schemes proposed in [LDB03] do not satisfy the semantic security w.r.t. the authority. This is obvious for the generic construction based on identity-based encryption which consists in only one flow of communication (since a scheme that achieves the strong security notions requires at least two flows). This is also true (to a lesser extent) for the RSA-based construction: for any third party, the semantic security relies (in the random oracle model) on the CDH assumption in a 2048-bit RSA group; but for the authority, it can be broken by solving two 1024-bit discrete logarithm problems. This task is much simpler in particular if the authority generates the RSA modulus $N = pq$ dishonestly (*e.g.* with $p - 1$ and $q - 1$ smooth). In order to make the scheme secure in our strong model, one needs (at least) to double the size of the RSA modulus and to make sure that the authority has selected and correctly employed a truly random seed in the generation of the RSA key pair [JG02].

G.3 An Efficient OSBE scheme

In this section, we present a high-level instantiation of OSBE with the previous primitives as black boxes. Thereafter, we provide a specific instantiation with linear ciphertexts. The overall security then relies on the DLin assumption, a quite standard assumption in the standard model. Its efficiency is of the same order of magnitude than the construction based on identity-based encryption [LDB03] (that only achieves weaker security notions) and better than the RSA-based scheme which provides similar security guarantees (in the random oracle model).

G.3.1 High-Level Instantiation

We assume we have an encryption scheme $\mathcal{E}$, a signature scheme $\mathcal{S}$ and a SPHF system onto a set $\mathbb{G}$. We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^\ell$ from a pseudo-random element v in $\mathbb{G}$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash-Lemma just leads to a security loss linear in the number of extractions. We describe an oblivious signature-based envelope system $\mathcal{OSBE}$, to send a private message $P \in \{0,1\}^\ell$:

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter:
  - it first generates the global parameters for the signature scheme (using SSetup), the encryption scheme (using ESetup), and the SPHF system (using $\mathsf{SPHFSetup}$);
  - it then generates the public key ek of the encryption scheme (using $\mathsf{EKeyGen}$, while the decryption key will not be used);

  The output param consists of all the individual param and the encryption key ek;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ runs $\mathsf{SKeyGen}(\mathsf{param})$ to generate a pair $(\mathsf{vk}, \mathsf{sk})$ of verification-signing keys;
- The $\mathsf{OSBESign}$ and $\mathsf{OSBEVerif}$ algorithms are exactly Sign and Verif from the signature scheme;
- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$: In the following, $\mathcal{L} = \mathcal{L}(\mathsf{vk}, M)$ will describe the language of the ciphertexts under the above encryption key ek of a valid signature of the input message M under the input verification key vk (hence vk and M as inputs, while param contains ek).
  - $\mathcal{R}$ generates and sends $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\mathcal{S}$ computes successively $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$; $\mathcal{S}$ sends hp, Q to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), c, r)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

G.3.2 Security Properties

Theorem G.3.1 [Correct] $\mathcal{OSBE}$ is **sound**.

Proof: Under the correctness of the SPHF system, $v' = v$, and thus $P' = (P \oplus \mathsf{KDF}(v)) \oplus \mathsf{KDF}(v') = P$. ∎

Theorem G.3.2 [Escrow-Free] $\mathcal{OSBE}$ is **escrow-free** if the encryption scheme $\mathcal{E}$ is *semantically secure*: $\mathsf{Adv}^{\mathsf{esc}}_{\mathcal{OSBE}}(k, t) \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k, t')$ with $t' \approx t$.

Proof: Let us assume $\mathcal{A}$ is an adversary against the escrow-free property of our scheme: The malicious adversary $\mathcal{A}$ is able to tell the difference between an interaction with $\mathcal{R}_0$ (who knows and uses a valid signature) and $\mathcal{R}_1$ (who does not use a valid signature), with advantage ε.

We now build an adversary $\mathcal{B}$ against the semantic security of the encryption scheme $\mathcal{E}$:

- $\mathcal{B}$ is first given the parameters for $\mathcal{E}$ and an encryption key ek;
- $\mathcal{B}$ emulates $\mathsf{OSBESetup}$: it runs SSetup and $\mathsf{SPHFSetup}$ by itself. For the encryption scheme $\mathcal{E}$, the parameters and the key have already been provided by the challenger of the encryption security game;
- $\mathcal{A}$ provides the verification key vk;
- $\mathcal{B}$ has to simulate all the oracles:

- $\mathtt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, $\mathcal{B}$ computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);
- $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$, for a message M and a valid signature σ: the adversary $\mathcal{B}$ outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
- $\mathtt{Exec}^*(\mathsf{vk}, M, \sigma, P)$: one first runs $\mathtt{Rec}(\mathsf{vk}, M, \sigma, 0)$ to generate c, that is provided to $\mathtt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).

- At some point, $\mathcal{A}$ outputs a message M and a valid signature σ, and $\mathcal{B}$ has to simulate $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, b)$: $\mathcal{B}$ sets $\sigma_0 \leftarrow \sigma$ and sets σ_1 as a random string. It sends (σ_0, σ_1) to the challenger of the semantic security of the encryption scheme and gets back c, an encryption of σ_β, for a random unknown bit β. It outputs c;
- $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b', that $\mathcal{B}$ forwards as its guess β' for the β involved in the semantic security game for $\mathcal{E}$.

Note that the above simulation perfectly emulates $\mathsf{Exp}^{\mathsf{esc}-\beta}_{\mathcal{OSBE},\mathcal{A}}(k)$ (since basically b is β, and b' is β'):

$$\varepsilon = \mathsf{Adv}^{\mathsf{esc}}_{\mathcal{OSBE},\mathcal{A}}(k) = \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{B}}(k) \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k,t).$$

∎

Theorem G.3.3 [Semantically Secure] $\mathcal{OSBE}$ is **semantically secure** if the signature is *unforgeable*, the SPHF is *smooth* and the encryption scheme is *semantically secure* (and under the pseudo-randomness of the KDF):

$$\mathsf{Adv}^{\mathsf{sem}}_{\mathcal{OSBE}}(k,t) \leq q_U\,\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k,t') + 2\,\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,q_S,t'') + 2\,\mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{SPHF}}(k) \text{with } t', t'' \approx t.$$

In the above formula, q_U denotes the number of interactions the adversary has with the sender, and q_S the number of signing queries the adversary asked.

Proof: Let us assume $\mathcal{A}$ is an adversary against the semantic security of our scheme: The malicious adversary $\mathcal{A}$ is able to tell the difference between an interaction with $\mathcal{S}_0$ (who owns P_0) and $\mathcal{S}_1$ (who owns P_1), with advantage ε. We start from this initial security game, and make slight modifications to bound ε.

Game $\mathcal{G}_0$. Let us emulate this security game:

- $\mathcal{B}$ emulates the initialization of the system: it runs $\mathsf{OSBESetup}$ by itself, and then the key generation algorithm $\mathsf{OSBEKeyGen}$ to generate $(\mathsf{vk}, \mathsf{sk})$;
- $\mathcal{B}$ has to simulate all the oracles:
  - $\mathtt{Sign}(\mathsf{vk}, M)$ and $\mathtt{Sign}^*(\mathsf{vk}, M)$: it runs the corresponding algorithm by itself;
  - $\mathtt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, $\mathcal{B}$ computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);
  - $\mathtt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ asks for a valid signature σ on M, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;

  - $\texttt{Exec}(\mathsf{vk}, M, P)$: one simply first runs $\texttt{Rec}(\mathsf{vk}, M, 0)$ to generate c, that is provided to $\texttt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).

- At some point, $\mathcal{A}$ outputs a message M and two inputs (P_0, P_1) to distinguish the sender, and $\mathcal{B}$ call back the above $\texttt{Send}(\mathsf{vk}, M, P_b)$ simulation to interact with $\mathcal{A}$;
- $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b'.

In this game, $\mathcal{A}$ has an advantage ε in guessing b:

$$\varepsilon = \Pr_{0}[b' = 1 | b = 1] - \Pr_{\mathcal{G}_0}[b' = 1 | b = 0] = 2 \times \Pr_{\mathcal{G}_0}[b' = b] - 1.$$

Game $\mathcal{G}_1^\beta$. This game involves the semantic security of the encryption scheme: $\mathcal{B}$ is already provided the parameters and the encryption key ek by the challenger of the semantic security of the encryption scheme, hence the initialization is slightly modified. In addition, $\mathcal{B}$ sets the bit $b = \beta$, and modifies the $\texttt{Rec}$ oracle simulation:

- $\texttt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ asks for a valid signature σ_0 on M, and sets σ_1 as a random string, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma_b; r)$.

Since $\mathcal{B}$ knows b, it finally outputs $\beta' = (b' = b)$.

Note that $\mathcal{G}_1^0$ is exactly $\mathcal{G}_0$, and the distance between $\mathcal{G}_1^0$ and $\mathcal{G}_1^1$ relies on the Left-or-Right security of the encryption scheme, which can be shown equivalent to the semantic security, with a lost linear in the number of encryption queries, which is actually the number q_U of interactions with a user (the sender in this case), due to the hybrid argument [BDJR97]:

$$\begin{aligned} q_U \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k) &\geq \Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1] \\ &= \Pr[b' = b | \beta = 0] - \Pr[b' = b | \beta = 1] \\ &= (2 \times \Pr_{\mathcal{G}_1^0}[b' = b] - 1) - (2 \times \Pr_{\mathcal{G}_1^1}[b' = b] - 1) \end{aligned}$$

As a consequence: $\varepsilon \leq q_U \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k) + (2 \times \Pr_{\mathcal{G}_1^1}[b' = b] - 1)$.

Game $\mathcal{G}_2$. This game involves the unforgeability of the signature scheme: $\mathcal{B}$ is already provided the parameters and the verification vk for the signature scheme, together with access to the signing oracle (note that all the signing queries Sign^* asked by the adversary in the $\texttt{FIND}$ stage, i.e., before the challenge interaction with $\texttt{Send}(\mathsf{vk}, M, P_b)$, are stored in $\mathcal{SM}$). The simulator $\mathcal{B}$ generates itself all the other parameters and keys, an namely the encryption key ek, together with the associated decryption key dk. For the $\texttt{Rec}$ oracle simulation, $\mathcal{B}$ keeps the random version (as in $\mathcal{G}_1^1$). In the challenge interaction with $\texttt{Send}(\mathsf{vk}, M, P_b)$, one stops the simulation and makes the adversary win if it uses a valid signature on a message $M \notin \mathcal{SM}$:

- $\texttt{Send}(\mathsf{vk}, M, P_b)$, during the challenge interaction: upon receiving c, if $\mathcal{M} \notin \mathcal{SM}$, it first decrypts c to get the input signature σ. If σ is a valid signature, one stops the game, sets $b' = b$ and outputs b'. If the signature is in not valid, the simulation remains unchanged;
- $\texttt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ sets σ as a random string, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$.

Because of the abort in the case of a valid signature on a new message, we know that the adversary cannot use such a valid signature in the challenge. So, since M should not be in $\mathcal{SM}$, the signature will be invalid. Actually, the unique difference from the previous game $\mathcal{G}_1^1$ is the abort in case of valid signature on a new message in the challenge phase, which probability is bounded by $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S)$. Using Shoup's Lemma [Sho02]:

$$\Pr_{\mathcal{G}_1^1}[b' = b] - \Pr_{\mathcal{G}_2}[b' = b] \leq \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S).$$

As a consequence: $\varepsilon \leq q_U \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k) + 2 \times \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S) + (2 \times \Pr_{\mathcal{G}_2}[b' = b] - 1)$.

Game $\mathcal{G}_3$. The last game involves the smoothness of the SPHF: The unique difference is in the computation of v in `Send` simulation, in the challenge phase only: $\mathcal{B}$ chooses a random $v \in \mathbb{G}$. Due to the statistical randomness of v in the previous game, in case the signature is not valid (a word that is not in the language), this game is statistically indistinguishable from the previous one:

$$\Pr_{\mathcal{G}_2}[b' = b] - \Pr_{\mathcal{G}_3}[b' = b] \leq \mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{SPHF}}(k).$$

Since P_b is now masked by a truly random value, no information leaks on b: $\Pr_{\mathcal{G}_3}[b' = b] = 1/2$. ∎

Theorem G.3.4 $\mathcal{OSBE}$ is **semantically secure w.r.t. the authority** if the SPHF is *pseudo-random* (and under the pseudo-randomness of the KDF):

$$\mathsf{Adv}^{\mathsf{sem}^*}_{\mathcal{OSBE}}(k, t) \leq 2 \times \mathsf{Adv}^{\mathsf{pr}}_{\mathcal{SPHF}}(k, t).$$

Proof: Let us assume $\mathcal{A}$ is an adversary against the semantic security w.r.t. the authority: The malicious adversary $\mathcal{A}$ is able to tell the difference between an eavesdropped interaction with $\mathcal{S}_0$ (who owns P_0) and $\mathcal{S}_1$ (who owns P_1), with advantage ε. We start from this initial security game, and make slight modifications to bound ε.

Game $\mathcal{G}_0$. Let us emulate this security game:

- $\mathcal{B}$ emulates the initialization of the system: it runs $\mathsf{OSBESetup}$ by itself;
- $\mathcal{A}$ provides the verification key vk;
- $\mathcal{B}$ has to simulate all the oracles:
  - $\mathtt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, $\mathcal{B}$ computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);
  - $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$, for a message M and a valid signature σ: $\mathcal{B}$ outputs the ciphertext $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\mathtt{Exec}^*(\mathsf{vk}, M, \sigma, P)$: one first runs $\mathtt{Rec}(\mathsf{vk}, M, \sigma, 0)$ to generate c, that is provided to $\mathtt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).
- At some point, $\mathcal{A}$ outputs a message M with a valid signature σ, and two inputs (P_0, P_1) to distinguish the sender, and $\mathcal{B}$ call back the above $\mathtt{Send}(\mathsf{vk}, M, P_b)$ and $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$ simulations to interact together and output the transcript $(c; \mathsf{hp}, Q)$;

- $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b'.

In this game, $\mathcal{A}$ has an advantage ε in guessing b:

$$\varepsilon = \Pr_{\mathcal{G}_0}[b' = 1 | b = 1] - \Pr_{\mathcal{G}_0}[b' = 1 | b = 0] = 2 \times \Pr_{\mathcal{G}_0}[b' = b] - 1.$$

Game $\mathcal{G}_1$. This game involves the pseudo-randomness of the SPHF: The unique difference is in the computation of v in `Send` simulation of the eavesdropped interaction, and so for the transcript: $\mathcal{B}$ chooses a random $v \in \mathbb{G}$ and computes $Q = P_b \oplus \mathsf{KDF}(v)$. Due to the pseudo-randomness of v in the previous game, since $\mathcal{A}$ does not know the random coins r used to encrypt σ, this game is computationally indistinguishable from the previous one.

$$\Pr_{\mathcal{G}_1}[b' = b] - \Pr_{\mathcal{G}_0}[b' = b] \leq \mathsf{Adv}^{\mathsf{pr}}_{\mathcal{SPHF}}(k, t).$$

Since P_b is now masked by a truly random value v, no information leaks on b: $\Pr_{\mathcal{G}_1}[b' = b] = 1/2$.

∎

G.3.3 Our Efficient OSBE Instantiation

Our first construction combines the linear encryption scheme [BBS04], the Waters signature scheme [Wat05] and a SPHF on linear ciphertexts [CS02, Sha07]. It thus relies on classical assumptions: CDH for the unforgeability of signatures and DLin for the semantic security of the encryption scheme. The formal definitions are recalled in the Appendix G.5.3.

Basic Primitives.

Given an encrypted Waters signature from the recipient, the sender is able to compute a projection key, and a hash corresponding to the expected signature, and send to the recipient the projection key and the product between the expected hash and the message P. If the recipient was honest (a correct ciphertext), it is able to compute the hash thanks to the projection key, and so to find P, in the other case it does not learn anything.

We briefly sketch the basic building blocks: linear encryption, Waters signature and the SPHF for linear tuples. They are more formally described in the appendix G.5.3.

All these primitives work in a pairing-friendly environment $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, where $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible bilinear map, for two groups $\mathbb{G}$ and $\mathbb{G}_T$, of prime order p, generated by g and $g_t = e(g, g)$ respectively.

Waters Signatures. The public parameters are a generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which defines the *Waters hash* of a message $M = (M_1, \ldots, M_k) \in \{0,1\}^k$ as $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. The public verification key is $\mathsf{vk} = g^z$, which corresponding secret signing key is $\mathsf{sk} = h^z$, for a random $z \xleftarrow{\$} \mathbb{Z}_p$. The signature on a message $M \in \{0,1\}^k$ is $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g^s)$, for some random $s \xleftarrow{\$} \mathbb{Z}_p$. It can be verified by checking $e(g, \sigma_1) = e(\mathsf{vk}, h) \cdot e(\mathcal{F}(M), \sigma_2)$. This signature scheme is *unforgeable* under the CDH assumption.

Linear Encryption. The secret key dk is a pair of random scalars (y_1, y_2) and the public key is $\mathsf{ek} = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$. One encrypts a message $M \in \mathbb{G}$ as $c = (c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1 + r_2} \cdot M)$, for random scalars $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. To decrypt, one computes $M = c_3 / (c_1^{1/y_1} c_2^{1/y_2})$. This encryption scheme is *semantically secure* under the DLin assumption.

DLin-compatible Smooth-Projective Hash Function. This is actually a weaker variant of [Sha07]. The language $\mathcal{L}$ consists of the linear tuples w.r.t. a basis (u, v, g). For a linear encryption key $\mathsf{ek} = (Y_1, Y_2)$, a ciphertext $C = (c_1, c_2, c_3)$ is an encryption of the message M if $(c_1, c_2, c_3/M)$ is a linear tuple w.r.t. the basis (Y_1, Y_2, g). The language $\mathsf{Lin}(\mathsf{ek}, M)$ consists of these ciphertexts. An SPHF for this language can be:

$$\begin{aligned}
\mathsf{HashKG}(\mathsf{Lin}(\mathsf{ek}, M)) &= \mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3 \\
\mathsf{Hash}(\mathsf{hk}; \mathsf{Lin}(\mathsf{ek}, M), C) &= c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3} \\
\mathsf{ProjKG}(\mathsf{hk}; \mathsf{Lin}(\mathsf{ek}, M), C) &= \mathsf{hp} = (Y_1^{x_1} g^{x_3}, Y_2^{x_2} g^{x_3}) \\
\mathsf{ProjHash}(\mathsf{hp}; \mathsf{Lin}(\mathsf{ek}, M), C; r) &= \mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}
\end{aligned}$$

This function is defined for linear tuples in $\mathbb{G}$, but it could work in any group, since it does not make use of pairings. And namely, we use it below in $\mathbb{G}_T$.

Smooth-Projective Hash Function for Linear Encryption of Valid Waters Signatures. We will consider a slightly more complex language: the ciphertexts under ek of a valid signature of M under vk. A given ciphertext $C = (c_1, c_2, c_3, \sigma_2)$ contains a valid signature of M if and only if (c_1, c_2, c_3) actually encrypts σ_1 such that (σ_1, σ_2) is a valid Waters signature on M. The latter means

$$(C_1 = e(c_1, g), C_2 = e(c_2, g), C_3 = e(c_3, g)/(e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2))$$

is a linear tuple in basis $(U = e(Y_1, g), V = e(Y_2, g), g_t = e(g, g))$ in $\mathbb{G}_T$. Since the basis consists of 3 elements of the form $e(\cdot, g)$, the projected key can be compacted in $\mathbb{G}$. We thus consider the language $\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)$ that contains these quadruples $(c_1, c_2, c_3, \sigma_2)$, and its SPHF:

$$\begin{aligned}
\mathsf{HashKG}(\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)) &= \mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3 \\
\mathsf{Hash}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) &= \\
& e(c_1, g)^{x_1} e(c_2, g)^{x_2} (e(c_3, g)/(e(h, \mathsf{vk}) e(\mathcal{F}(M), \sigma_2)))^{x_3} \\
\mathsf{ProjKG}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) &= \mathsf{hp} = (\mathsf{ek}_1^{x_1} g^{x_3}, \mathsf{ek}_2^{x_2} g^{x_3}) \\
\mathsf{ProjHash}(\mathsf{hp}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C; r) &= e(\mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}, g)
\end{aligned}$$

Instantiation.

We now define our OSBE protocol, where a sender $\mathcal{S}$ wants to send a private message $P \in \{0,1\}^\ell$ to a recipient $\mathcal{R}$ in possession of a Waters signature on a message M.

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter, defines a pairing-friendly environment $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, the public parameters $h \xleftarrow{\$} \mathbb{G}$, an encryption key $\mathsf{ek} = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$, where $(y_1, y_2) \xleftarrow{\$} \mathbb{Z}_p^2$, and $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ for the Waters signature. All these elements constitute the string param;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$, the authority generates a pair of keys $(\mathsf{vk} = g^z, \mathsf{sk} = h^z)$ for a random scalar $z \xleftarrow{\$} \mathbb{Z}_p$;
- $\mathsf{OSBESign}(\mathsf{sk}, M)$ produces a signature $\sigma = (h^z \mathcal{F}(M)^s, g^s)$;
- $\mathsf{OSBEVerif}(\mathsf{vk}, M, \sigma)$ checks if $e(\sigma_1, g) = e(\sigma_2, \mathcal{F}(M)) \cdot e(h, \mathsf{vk})$.

- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ runs as follows:
  - $\mathcal{R}$ chooses random $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and sends a linear encryption of σ: $C = (c_1 = \mathsf{ek}_1^{r_1}, c_2 = \mathsf{ek}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1, \sigma_2)$
  - $\mathcal{S}$ chooses random $x_1, x_2, x_3 \xleftarrow{\$} \mathbb{Z}_p^3$ and computes:
    - $\mathsf{HashKG}(\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)) = \mathsf{hk} = (x_1, x_2, x_3)$;
    - $\mathsf{Hash}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) = v = e(c_1, g)^{x_1} e(c_2, g)^{x_2} (e(c_3, g)/(e(h, \mathsf{vk}) e(\mathcal{F}(M), \sigma_2)))^{x_3}$;
    - $\mathsf{ProjKG}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) = \mathsf{hp} = (\mathsf{ek}_1^{x_1} g^{x_3}, \mathsf{ek}_2^{x_2} g^{x_3})$.
  - $\mathcal{S}$ then sends $(\mathsf{hp}, Q = P \oplus \mathsf{KDF}(v))$ to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = e(\mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}, g)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

An asymmetric instantiation can be found in the Appendix G.7.2.

G.3.4 Security and Efficiency

We now provide a security analysis of this scheme. This instantiation differs, from the high-level instantiation presented before, in the ciphertext C of the signature $\sigma = (\sigma_1, \sigma_2)$. The second half of the signature indeed remains in clear. It thus does not guarantee the semantic security on the signature used in the ciphertext. However, granted Waters signature randomizability, one can re-randomize the signature each time, and thus provide a totally new σ_2: it does not leak any information about the original signature. The first part of the ciphertext (c_1, c_2, c_3) does not leak any additional information under the DLin assumption. As a consequence, the global ciphertext guarantees the semantic security of the original signature if a new re-randomized signature is encrypted each time. We can now apply the high-level construction security, and all the assumptions hold under the DLin one:

Theorem G.3.5 Our OSBE scheme is secure (i.e., escrow-free, semantically secure, and semantically secure w.r.t. the authority) under the DLin assumption (and the pseudo-random generator in the KDF).

Our proposed scheme needs one communication for $\mathcal{R}$ and one for $\mathcal{S}$, so it is round-optimal. Communication also consists of few elements, $\mathcal{R}$ sends 4 group elements, and $\mathcal{S}$ answers with 2 group elements only and an ℓ-bit string for the masked $P \in \{0,1\}^\ell$. As explained in Remark G.2.2, this has to be compared with the RSA-based scheme from [LDB03] which requires 2 elements in RSA groups (with double-length modulus). For a 128-bit security level, using standard Type-I bilinear groups implementation [GPS08], we obtain a 62.5% improvement[1] in communication complexity over the RSA-based scheme proposed in the original paper [LDB03].

While reducing the communication cost of the scheme, we have improved its security and it now fits the proposed applications. In [LDB03], such schemes were proposed for applications where someone wants to transmit a confidential information to an agent belonging to a specific agency. However the agent does not want to give away his signature. As they do not consider eavesdropping and replay in their semantic security nothing prevents an adversary to replay a part of a previous interaction to impersonate a CIA agent (to recall their example). In practice, an additional secure communication channel, such as with SSL, was required in their security model, hence increasing the communication cost: our protocol is secure by itself.

[1] The improvement is even more important for the scheme described in Appendix G.7.2 since, using standard Type-II or Type-III bilinear groups, the communication complexity is only 3/16-th of the one of the RSA-based scheme.

G.4 An efficient Blind Signature

G.4.1 Definitions

A more formal definition of blind signatures is provided in the Appendix G.6, but we briefly recall it in this section: A blind signature scheme $\mathcal{BS}$ is defined by a setup algorithm $\mathsf{BSSetup}(1^k)$ that generates the global parameters param, and key generation algorithm $\mathsf{BSKeyGen}(\mathsf{param})$ that outputs a pair $(\mathsf{vk}, \mathsf{sk})$, and interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$ which provides $\mathcal{U}$ with a signature on m, and a verification algorithm $\mathsf{BSVerif}(\mathsf{vk}, m, \sigma)$ that checks its validity. The security of a blind signature scheme is defined through the unforgeability and blindness properties: An adversary against the unforgeability tries to generate $q_s + 1$ valid message-signature pairs after at most q_s complete interactions with the honest signer; The blindness condition states that a malicious signer should be unable to decide which of two messages m_0, m_1 has been signed first in two executions with an honest user.

G.4.2 Our Instantiation

We now present a new way to obtain a blind signature scheme in the standard model under classical assumptions with a common-reference string. This is an improvement over [BFPV11]. We are going to use the same building blocks as before, so linear encryption, Waters signatures and a SPHF on linear ciphertexts. More elaborated languages will be required, but just conjunctions and disjunctions of classical languages, as done in [ACP09] (see Appendix G.5.2 and G.5.4), hence the efficient construction. Our blind signature scheme is defined by:

- $\mathsf{BSSetup}(1^k)$, where k is the security parameter, generates a pairing-friendly system $(p, \mathbb{G}, g, \mathbb{G}_T, e)$ and an encryption key $\mathsf{ek} = (u, v, g) \in \mathbb{G}^3$. It also chooses at random $h \in \mathbb{G}$ and generators $\vec{u} = (u_i)_{i \in [\![1,\ell]\!]} \in \mathbb{G}^\ell$ for the Waters function. It outputs the global parameters $\mathsf{param} = (p, \mathbb{G}, g, \mathbb{G}_T, e, \mathsf{ek}, h, \vec{u})$;
- $\mathsf{BSKeyGen}(\mathsf{param})$ picks at random a secret key $\mathsf{sk} = x$ and computes the verification key $\mathsf{vk} = g^x$;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on M
  - $\mathcal{U}$ computes the bit-per-bit encryption of M by encrypting each $u_i^{M_i}$ in b_i, $\forall i \in [\![1, \ell]\!], b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i}; (r_{i,1}, r_{i,2})) = (u^{r_{i,1}}, v^{r_{i,2}}, g^{r_{i,1}+r_{i,2}} u_i^{M_i})$. Then writing $r_1 = \sum r_{i,1}$ and $r_2 = \sum r_{i,2}$, he computes the encryption c of $\mathsf{vk}^{r_1+r_2}$ with
    $$\mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}^{r_1+r_2}; (s_1, s_2)) = (u^{s_1}, v^{s_2}, g^{s_1+s_2} \mathsf{vk}^{r_1+r_2}).$$
    $\mathcal{U}$ then sends $(c, (b_i))$;
  - On input of these ciphertexts, the algorithm $\mathcal{S}$ computes the corresponding SPHF, considering the language $\mathcal{L}$ of valid ciphertexts. This is the conjunction of several languages (see Appendix G.5.4 for details:)
    1. One checking that each b_i encrypts a bit in basis u_i: in $\mathsf{BLin}(\mathsf{ek}, u_i)$;
    2. One considering $(d_1, d_2, c_1, c_2, c_3)$, that checks if (c_1, c_2, c_3) encrypts an element d_3 such that (d_1, d_2, d_3) is a linear tuple in basis (u, v, vk): in $\mathsf{ELin}(\mathsf{ek}, \mathsf{vk})$, where $d_1 = \prod_i b_{i,1}$ and $d_2 = \prod_i b_{i,2}$.
  - $\mathcal{S}$ computes the corresponding Hash-value v, extracts $K = \mathsf{KDF}(v) \in \mathbb{Z}_p$, generates the blinded signature $(\sigma_1'' = h^x \delta^s, \sigma_2' = g^s)$, where $\delta = u_0 \prod_i b_{i,3} = \mathcal{F}(M) g^{r_1+r_2}$, and sends $(\mathsf{hp}, Q = \sigma_1'' \times g^K, \sigma_2')$;

  - Upon receiving $(\mathsf{hp}, Q, \sigma'_2)$, using its witnesses and hp, $\mathcal{U}$ computes the $\mathsf{ProjHash}$-value v', extracts $K' = \mathsf{KDF}(v')$ and unmasks $\sigma''_1 = Q \times g^{-K'}$. Thanks to the knowledge of r_1 and r_2, it can compute $\sigma'_1 = \sigma''_1 \times (\sigma'_2)^{-r_1-r_2}$. Note that if $v' = v$, then $\sigma'_1 = h^x \mathcal{F}(M)^s$, which together with $\sigma'_2 = g^s$ is a valid Waters signature on M. It can thereafter re-randomize the final signature $\sigma = (\sigma'_1 \cdot \mathcal{F}(M)^{s'}, \sigma'_2 \cdot g^{s'})$.

- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$, checks whether $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

The idea is to remove any kind of proof of knowledge in the protocol, which was the main concern in [BFPV11], and use instead a SPHF. This way, we obtain a protocol where the user first sends $3\ell + 6$ group elements for the ciphertext, and receives back $5\ell + 4$ elements for the projection key and 2 group elements for the blinded signature. So $8\ell + 12$ group elements are used in total. This has to be compared to $9\ell + 24$ in [BFPV11]. We both reduce the linear and the constant parts in the number of group elements involved while relying on the same hypotheses. And the final result is still a standard Waters signature.

Remark G.4.1 In [GRS+11], Garg *el al.* proposed the first round-optimal blind signature scheme in the standard model, without CRS. In order to remove the CRS, their scheme makes use of ZAPs [DN07] and is quite inefficient. Moreover, its security relies on a stronger assumption (namely, sub-exponential hardness of one-to-one one-way functions). A natural idea is to replace the CRS in our scheme with Groth-Ostrovsky-Sahai ZAP [GOS06a] based on the DLin assumption. This change would only double the communication complexity, but we do not know how to prove the security of the resulting scheme[2]. It remains a tantalizing open problem to design an efficient round-optimal blind signature in the standard model without CRS.

G.4.3 Security

In blind signatures, one expects two kinds of security properties:

- *blindness*, preventing the signer to be able to recognize which message was signed during a specific interaction. Due to Waters re-randomizability and linear encryption, this property is guaranteed in our scheme under the DLin assumption;

- *unforgeability*, guaranteeing the user will not be able to output more signed messages than the number of actual interactions. In this scheme, granted the extractability of the encryption (the simulator can know the decryption key) one can show that the user cannot provide a signature on a message different from the ones it asked to be blindly signed. Hence, the unforgeability relies on the Waters unforgeability, that is the CDH assumption.

Theorem G.4.2 Our blind signature scheme is blind[3] under the DLin assumption (and the pseudo-randomness of the KDF) and unforgeable under the CDH assumption.

A full proof can be found in appendix G.6.

[2] Indeed, opening the commitment scheme in the ZAP and forging a signature relies on the same computational assumption, which makes it impossible to apply the complexity leveraging argument from [GRS+11].

[3] Our scheme satisfies the *blindness against covert adversaries* security notion. It formalizes the security desired for most applications of blind signatures (*e.g.* e-cash or e-voting). Covert adversaries have the property that they may deviate arbitrarily from the protocol specification in an attempt to cheat, but do not wish to be "caught" doing so.

Acknowledgments

This work was supported by the French ANR-07-TCOM-013-04 PACE Project, by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II.

G.5 Formal Definitions

G.5.1 Formal Definitions of the Primitives

Encryption scheme. An encryption scheme is defined by four algorithms

$$(\mathsf{ESetup}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}) :$$

- $\mathsf{ESetup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{EKeyGen}(\mathsf{param})$ generates a pair of keys, the public (encryption) key ek and the private (decryption) key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ outputs the plaintext m encrypted in c.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$ output by $\mathsf{EKeyGen}(\mathsf{param})$ and all messages m we have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m)) = m$.
- *Indistinguishability under chosen-plaintext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls `FIND` and `GUESS`.

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{ESetup}(1^k)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{EKeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek})$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*)$
6. `RETURN` b'

The advantages are

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(k) &= \Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(k) = 1] \\ \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k,t) &= \max_{\mathcal{A} \leq t} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(k). \end{aligned}$$

Signature scheme. A signature scheme is defined by four algorithms

$$(\mathsf{SSetup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verif}) :$$

- $\mathsf{SSetup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{SKeyGen}(\mathsf{param})$ generates a pair of keys, the public (verification) key vk and the private (signing) key sk;

- $\mathsf{Sign}(\mathsf{sk}, m; s)$ produces a signature σ on the input message m, under the signing key sk, and using the random coins s;
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

A signature scheme $\mathcal{S}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{vk}, \mathsf{sk})$ and all messages m we have $\mathsf{Verif}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$.
- *Existential unforgeability under (adaptive) chosen-message attacks*: this security notion can be formalized by the following security game, where it makes use of the oracle `Sign`:
  - $\mathtt{Sign}(\mathsf{sk}, m)$: This oracle outputs a valid signature on m under the signing key sk. The input queries m are added to the list $\mathcal{SM}$.

  The success probabilities are

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{SSetup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathtt{Sign}(\mathsf{sk}, \cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. `IF` $M \in \mathcal{SM}$ `RETURN` 0
6. `ELSE RETURN` b

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k) = 1] \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k).$$

Smooth Projective Hash Function. An SPHF over a language $\mathcal{L} \subset X$, onto a set $\mathbb{G}$, is defined by five algorithms $(\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$:

- $\mathsf{SPHFSetup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathcal{L}$;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$ generates a hashing key hk;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ generates the projection key hp, possibly depending on the word W [GL03, ACP09] from the hashing key;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ outputs the hash value $v \in \mathbb{G}$, on W from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$ outputs the hash value $v' \in \mathbb{G}$, on W from the projection key and the witness.

A Smooth Projective Hash Function $\mathcal{SPHF}$ should satisfy the following properties:

- *Correctness*: Let $W \in \mathcal{L}$ and w a witness of this membership. Then, for all hash keys hk and projected hash keys hp we have $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, c)$.
- *Smoothness*: For all $W \in X \setminus \mathcal{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{SPHFSetup}(1^k), \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{SPHFSetup}(1^k), \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{G} \end{array} \right\}.$$

This is formalized by

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{SPHF}}(k) = \sum_{V \in \mathbb{G}} \left| \Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V] \right| \text{ is negligible.}$$

- *Pseudo-Randomness*: If $c \in \mathcal{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathcal{SPHF},\mathcal{A}}(k) = \Pr_{\Delta_1}[\mathcal{A}(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]$$

is negligible.

G.5.2 Operations on Smooth Projective Hash Functions

We recall the constructions of SPHF on disjunctions and conjunctions of languages [ACP09]. Let us assume we have two Smooth Projective Hash Functions, defined by $\mathcal{SPHF}_1$ and $\mathcal{SPHF}_2$, on two languages, $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, both subsets of X, with hash values in the same group $(\mathbb{G}, \oplus)$. We note W an element of X, w_i a witness that $W \in \mathcal{L}_i$, $\mathsf{hk}_i = \mathsf{HashKG}_i(\mathcal{L}_i, \mathsf{param})$ and $\mathsf{hp}_i = \mathsf{ProjKG}_i(\mathsf{hk}_i, (\mathcal{L}_i, \mathsf{param}_i), W)$.

We can then define the SPHF on $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$, where $w = (w_1, w_2)$ as:

- $\mathsf{SPHFSetup}(1^k)$, $\mathsf{param} = (\mathsf{param}_1, \mathsf{param}_2)$, and $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$: $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: $\mathsf{Hash}_1(\mathsf{hk}_1, (\mathcal{L}_1, \mathsf{param}_1), W) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, (\mathcal{L}_2, \mathsf{param}_2), W)$
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w = (w_1, w_2))$:

$$\mathsf{ProjHash}_1(\mathsf{hp}_1, (\mathcal{L}_1, \mathsf{param}_1), W, w_1) \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2, (\mathcal{L}_2, \mathsf{param}_2), W, w_2)$$

We can also define the SPHF on $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$, where $w = w_1$ or $w = w_2$ as:

- $\mathsf{SPHFSetup}(1^k)$, $\mathsf{param} = (\mathsf{param}_1, \mathsf{param}_2)$, and $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$: $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta)$ where

$$\mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1, (\mathcal{L}_1, \mathsf{param}_1), W) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, (\mathcal{L}_2, \mathsf{param}_2), W)$$

- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: $\mathsf{Hash}_1(\mathsf{hk}_1, (\mathcal{L}_1, \mathsf{param}_1), W)$
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$: If $W \in \mathcal{L}_1$, $\mathsf{ProjHash}_1(\mathsf{hp}_1, (\mathcal{L}_1, \mathsf{param}_1), W, w_1)$, else (if $W \in \mathcal{L}_2$), $\mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2, (\mathcal{L}_2, \mathsf{param}_2), W, w_2)$

G.5.3 Our Concrete Primitives

In the following, we consider two different pairing-friendly settings;

- *Symmetric bilinear structure:* $(p, \mathbb{G}, g, \mathbb{G}_T, e)$ that gives the description of two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with generators g and $e(g, g)$ respectively where e is an efficiently computable non-degenerate bilinear map.
- *Asymmetric bilinear structure:* $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$ that gives the description of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p with generators g_1, g_2 and $e(g_1, g_2)$ respectively where e is an efficiently computable non-degenerate bilinear map.

Linear encryption (in a symmetric structure). The Linear encryption scheme was introduced by Boneh, Boyen and Shacham in [BBS04]:

- $\mathsf{ESetup}(1^k)$: the global parameters param consist of the description of a symmetric bilinear structure $(p, \mathbb{G}, g, \mathbb{G}_T, e)$;
- $\mathsf{EKeyGen}(\mathsf{param})$ picks a pair of random scalars $(y_1, y_2) \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{ek} = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$, and the secret key as $\mathsf{dk} = (y_1, y_2)$;
- $\mathsf{Encrypt}(\mathsf{ek}, M)$ on input a message $M \in \mathbb{G}$, it picks at random $r_1, r_2 \in \mathbb{Z}_p$ and computes $c_1 = Y_1^{r_1}$, $c_2 = Y_2^{r_2}$, $c_3 = g^{r_1+r_2} \cdot M$. It outputs the ciphertext $c = (c_1, c_2, c_3)$;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ on input a ciphertext $c = (c_1, c_2, c_3)$, it outputs $M = c_3/(c_1^{1/y_1} c_2^{1/y_2})$.

This scheme is semantically secure against chosen-plaintext attacks under the DLin assumption:

Definition G.5.1 [Decision Linear assumption (DLin)] Let $\mathbb{G}$ be a cyclic group of prime order p. The DLin assumption states that given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$ for random scalars $a, b, x, y, c \in \mathbb{Z}_p$, it is hard to decide whether $c = a + b$.

When $(g, u = g^x, v = g^y)$ is fixed, a tuple (u^a, v^b, g^{a+b}) is called a linear tuple w.r.t. (u, v, g), whereas a tuple (u^a, v^b, g^c) for a random and independent c is called a random tuple.

ElGamal encryption (in an asymmetric structure). In asymmetric structures, the DDH assumption can hold, one can thus use the ElGamal encryption:

- $\mathsf{ESetup}(1^k)$: the global parameters param consist of the description of an asymmetric bilinear structure $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$;
- $\mathsf{EKeyGen}(\mathsf{param})$ picks a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{ek} = g^y$, and the secret key as $\mathsf{dk} = y$;
- $\mathsf{Encrypt}(\mathsf{ek}, M)$ on input a message $m \in \mathbb{G}_1$, it picks at random $r \in \mathbb{Z}_p$ and computes $c_1 = g^r$ and $c_2 = \mathsf{ek}^r \cdot m$. It outputs the ciphertext $c = (c_1, c_2)$;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ on input a ciphertext $c = (c_1, c_2)$, it outputs $m = c_2/c_1^y$.

This scheme is semantically secure against chosen-plaintext attacks under the DDH assumption in $\mathbb{G}_1$:

Definition G.5.2 [Decisional Diffie-Hellman Assumption (DDH)] In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$. The DDH assumption in $\mathbb{G}_i$ states that given $(g_i, g_i^a, g_i^b, g_i^c) \in \mathbb{G}_i$, it is hard to determine whether $c = ab$ for random scalars $a, b, c \in \mathbb{Z}_p$.

Waters signature (in a symmetric structure). The original Waters Signature has been proposed in [Wat05]:

- $\mathsf{Setup}(1^k)$: in a symmetric bilinear structure $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, one chooses a vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We also need an extra generator $h \xleftarrow{\$} \mathbb{G}$. The global parameters param consist of all these elements $(p, \mathbb{G}, g, \mathbb{G}_T, e, h, \vec{u})$.
- $\mathsf{SKeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = g^x$, and the secret key as $\mathsf{sk} = h^x$.

- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g^s)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the CDH assumption in $\mathbb{G}$:

Definition G.5.3 [Computational Diffie-Hellman assumption (CDH)] Let $\mathbb{G}$ be a cyclic group of prime order p. The CDH assumption in $\mathbb{G}$ states that for a generator g of $\mathbb{G}$ and random $a, b \in \mathbb{Z}_p$, given (g, g^a, g^b) it is hard to compute g^{ab}.

Waters signature (in an asymmetric structure). An asymmetric variant of Waters signatures has been proposed in [BFPV11]:

- $\mathsf{Setup}(1^k)$: in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$, one chooses a random vector $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We also need an extra generator $h_1 \xleftarrow{\$} \mathbb{G}_1$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, \vec{u})$.
- $\mathsf{SKeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = g_2^x$, and the secret key as $\mathsf{sk} = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g_1^s, \sigma_3 = g_2^s)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g_2) = e(h_1, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_3)$, and $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$:

Definition G.5.4 [The Advanced Computational Diffie-Hellman problem (CDH^+)] In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$. The CDH^+ assumption states that given a 5-tuple $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

G.5.4 Our Smooth Projective Hash Functions

In this subsection, we present the languages we use in our first instantiations of OSBE and Blind Signatures.

Linear Language. In the following, we will denote $\mathsf{Lin}(\mathsf{ek}, M)$ the language of the linear encryptions C of the message M under the encryption key $\mathsf{ek} = (Y_1, Y_2)$. Clearly, for $M = 1_{\mathbb{G}}$, the language contains the linear tuples in basis (Y_1, Y_2, g). The SPHF system is defined by, for $\mathsf{ek} = (Y_1, Y_2)$ and $C = (c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1+r_2} \times M)$

$$\mathsf{HashKG}(\mathsf{Lin}(\mathsf{ek}, M)) = \mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3 \qquad \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), C) = c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3}$$
$$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), C) = \mathsf{hp} = (Y_1^{x_1} g^{x_3}, Y_2^{x_2} g^{x_3}) \qquad \mathsf{ProjHash}(\mathsf{hp}, \mathsf{Lin}(\mathsf{ek}, M), C, r) = \mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}$$

Theorem G.5.5 This Smooth Projective Hash Function is correct.

Proof: With the above notations:

- $\mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), C) = c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3} = Y_1^{r_1 x_1} Y_2^{r_2 x_2} g^{(r_1+r_2)x_3}$

- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{Lin}(\mathsf{ek}, M), C, r) = \mathsf{hp}_1^{r_1}\mathsf{hp}_2^{r_2} = (Y_1^{x_1}g^{x_3})^{r_1}(Y_2^{x_2}g^{x_3})^{r_2} = Y_1^{r_1x_1}Y_2^{r_2x_2}g^{(r_1+r_2)x_3}$

∎

Theorem G.5.6 This Smooth Projective Hash Function is smooth.

Proof: Let us show that from an information theoretic point of view, $v = \mathsf{Hash}(\mathsf{hk}, \mathcal{L}(\mathsf{ek}, M), C)$ is unpredictable, even knowing hp, when C is not a correct ciphertext: $C = (c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_3} \times M)$, for $r_3 \neq r_1 + r_2$. We recall that $\mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), C) = Y_1^{r_1x_1}Y_2^{r_2x_2}g^{r_3x_3}$ and $\mathsf{hp} = (Y_1^{x_1}g^{x_3}, Y_2^{x_2}g^{x_3})$: If we denote $Y_1 = g^{y_1}$ and $Y_2 = g^{y_2}$, we have:

$$\begin{pmatrix} \log \mathsf{hp}_1 \\ \log \mathsf{hp}_2 \\ \log v \end{pmatrix} = \begin{pmatrix} y_1 & 0 & 1 \\ 0 & y_2 & 1 \\ y_1r_1 & y_2r_2 & r_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

The determinant of this matrix is $y_1y_2(r_3 - r_1 - r_2)$, which is non-zero if C does not belong to the language ($r_3 \neq r_1 + r_2$). So v is independent from hp and C.∎

Theorem G.5.7 This Smooth Projective Hash Function is pseudo-random under the DLin assumption (the semantic security of the Linear encryption).

Proof: As shown above, when c encrypts $M' \neq M$, then the distributions

$$\begin{aligned} \mathcal{D}_1 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \\ \mathcal{D}_2 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), c)\} \end{aligned}$$

are perfectly indistinguishable. Under the semantic security of the Linear encryption, $\mathcal{E}_{\mathsf{ek}}(M)$ and $\mathcal{E}_{\mathsf{ek}}(M')$ are computationally indistinguishable, and so are the distributions

$$\begin{aligned} \mathcal{D}_0 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \\ \mathcal{D}_1 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \end{aligned}$$

and the distributions

$$\begin{aligned} \mathcal{D}_2 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), c)\} \\ \mathcal{D}_3 &= \{\mathsf{Lin}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), c)\} \end{aligned}$$

As a consequence, $\mathcal{D}_0$ and $\mathcal{D}_3$ are computationally indistinguishable, which proves the result. ∎

Bit Encryption Language. In our blind signature protocol, we need to "prove" that a ciphertext encrypts a bit in exponent of a basis u_i. That is the language $\mathsf{BLin}(\mathsf{ek}, u_i) = \mathsf{Lin}(\mathsf{ek}, 1_{\mathbb{G}}) \cup \mathsf{Lin}(\mathsf{ek}, u_i)$. This is thus a simple disjunction of two $\mathcal{SPHF}$:

- $\mathsf{HashKG}(\mathsf{BLin}(\mathsf{ek}, u_i))$: $\mathsf{hk} = ((x_1, x_2, x_3), (y_1, y_2, y_3)) \xleftarrow{\$} \mathbb{Z}_p^6$
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{BLin}(\mathsf{ek}, u_i), W)$: $\mathsf{hp} = ((Y_1^{x_1}g^{x_3}, Y_2^{x_2}g^{x_3}), (Y_1^{y_1}g^{y_3}, Y_2^{y_2}g^{y_3}), \mathsf{hp}_\Delta)$ where

$$\mathsf{hp}_\Delta = c_1^{x_1}c_2^{x_2}(c_3)^{x_3} \cdot c_1^{y_1}c_2^{y_2}(c_3/u_i)^{y_3}$$

- $\mathsf{Hash}(\mathsf{hk}, \mathsf{BLin}(\mathsf{ek}, u_i), W)$: $v = c_1^{x_1} c_2^{x_2} c_3^{x_3}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{BLin}(\mathsf{ek}, u_i), W, w)$: If $W \in \mathcal{L}_1$, $v' = \mathsf{hp}_{1,1}^{r_1} \cdot \mathsf{hp}_{1,2}^{r_2}$,
  else (if $W \in \mathcal{L}_2$), $v' = \mathsf{hp}_\Delta / \mathsf{hp}_{2,1}^{r_1} \cdot \mathsf{hp}_{2,2}^{r_2}$

The correctness, smoothness and pseudo-randomness properties of such function directly follow from those of the SPHF on $\mathsf{Lin}(\mathsf{pk}, 1_{\mathbb{G}})$ and $\mathsf{Lin}(\mathsf{pk}, u_i)$. Each projection key is composed of 5 group elements.

Encrypted Linear Language. We also need to consider a language $\mathsf{ELin}(\mathsf{ek}, \mathsf{vk})$ of tuples $(d_1, d_2, c_1, c_2, c_3)$, where (c_1, c_2, c_3) encrypts d_3 under the public key $\mathsf{ek} = (u, v)$, such that (d_1, d_2, d_3) is a linear tuple in basis (u, v, vk). This can also be expressed as $c_3 = \alpha \times d_3$, where d_3 is the plaintext in (c_1, c_2, c_3) under ek, which means that (c_1, c_2, α) is a linear tuple in basis (u, v, g), and (d_1, d_2, d_3) should be a linear tuple in basis (u, v, vk).

More concretely, we consider words $W = (d_1 = u^{r_1}, d_2 = v^{r_2}, c_1 = u^{s_1}, c_2 = v^{s_2}, c_3 = g^{s_1+s_2} \cdot \mathsf{vk}^{r_1+r_2})$, with witness $w = (r_1, r_2, s_1, s_2)$. We have $\alpha = g^{s_1+s_2}$ and $d_3 = \mathsf{vk}^{r_1+r_2}$, but they should remain secret, which requires a specific function, and not a simple conjunction of languages:

- $\mathsf{HashKG}(\mathsf{ELin}(\mathsf{ek}, \mathsf{vk}))$: $\mathsf{hk} = (x_1, x_2, x_3, x_4, x_5)$
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{ELin}(\mathsf{ek}, \mathsf{vk}), W)$: $\mathsf{hp} = (u^{x_1} g^{x_5}, v^{x_2} g^{x_5}, u^{x_3} g^{x_5}, v^{x_4} g^{x_5})$
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{ELin}(\mathsf{ek}, \mathsf{vk}), W)$: $v = e(d_1, \mathsf{vk})^{x_1} \cdot e(d_2, \mathsf{vk})^{x_2} \cdot e(c_1, g)^{x_3} \cdot e(c_2, g)^{x_4} \cdot e(c_3, g)^{x_5}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{ELin}(\mathsf{ek}, \mathsf{vk}), W, w)$: $v' = e(\mathsf{hp}_1, \mathsf{vk})^{r_1} \cdot e(\mathsf{hp}_2, \mathsf{vk})^{r_2} \cdot e(\mathsf{hp}_3, g)^{s_1} \cdot e(\mathsf{hp}_4, g)^{s_2}$

We now study the security of this SPHF:

Theorem G.5.8 This Smooth Projective Hash Function is correct.

Proof: With the above notations:

$$\begin{aligned}
v &= e(d_1, \mathsf{vk})^{x_1} \cdot e(d_2, \mathsf{vk})^{x_2} \cdot e(c_1, g)^{x_3} \cdot e(c_2, g)^{x_4} \cdot e(c_3, g)^{x_5}) \\
&= e(u^{\mathsf{sk} r_1 x_1}, g) \cdot e(v^{\mathsf{sk} r_2 x_2}, g) \cdot e(u^{s_1 x_3}, g) \cdot e(v^{s_2 x_4}, g) \cdot e(g^{(\mathsf{sk}(r_1+r_2)+(s_1+s_2))x_5}, g) \\
&= e(u^{\mathsf{sk} r_1 x_1 + s_1 x_3}, g) \cdot e(v^{\mathsf{sk} r_2 x_2 + s_2 x_4}, g) \cdot e(g^{(\mathsf{sk}(r_1+r_2)+(s_1+s_2))x_5}, g) \\
v' &= e(\mathsf{hp}_1, \mathsf{vk})^{r_1} \cdot e(\mathsf{hp}_2, \mathsf{vk})^{r_2} \cdot e(\mathsf{hp}_3, g)^{s_1} \cdot e(\mathsf{hp}_4, g)^{s_2} \\
&= e(u^{\mathsf{sk} r_1 x_1} g^{\mathsf{sk} r_1 x_5}, g) \cdot e(v^{\mathsf{sk} r_2 x_2} g^{\mathsf{sk} r_2 x_5}, g) \cdot e(u^{s_1 x_3} g^{s_1 x_5}, g) \cdot e(v^{s_2 x_4} g^{\mathsf{sk} s_2 x_5}, g) \\
&= e(u^{\mathsf{sk} r_1 x_1 + s_1 x_3}, g) \cdot e(v^{\mathsf{sk} r_2 x_2 + s_2 x_4}, g) e(g^{(\mathsf{sk}(r_1+r_2)+(s_1+s_2))x_5}, g)
\end{aligned}$$

∎

Theorem G.5.9 This Smooth Projective Hash Function is smooth.

Proof: Let us show that from an information theoretic point of view, v is unpredictable, even knowing hp, when W is not in the language: $W = (d_1 = u^{r_1}, d_2 = v^{r_2}, c_1 = u^{s_1}, c_2 = v^{s_2}, c_3 = g^t \cdot \mathsf{vk}^{r_1+r_2})$, for $t \neq s_1 + s_2$. We recall that

$$v = e(u^{\mathsf{sk} r_1 x_1 + s_1 x_3}, g) \cdot e(v^{\mathsf{sk} r_2 x_2 + s_2 x_4}, g) \cdot e(g^{(\mathsf{sk}(r_1+r_2)+(s_1+s_2))x_5}, g) = e(H, g)$$

for

$$H = u^{\mathsf{sk}r_1x_1+s_1x_3} \cdot v^{\mathsf{sk}r_2x_2+s_2x_4} \cdot g^{(\mathsf{sk}(r_1+r_2)+(s_1+s_2))x_5}$$

and

$$\mathsf{hp} = ((u^{x_1}g^{x_5}, v^{x_2}g^{x_5}), (u^{x_3}g^{x_5}, v^{x_4}g^{x_5}))$$

If we denote $u = g^{y_1}$ and $v = g^{y_2}$, we have:

$$\begin{pmatrix} \log \mathsf{hp}_1 \\ \log \mathsf{hp}_2 \\ \log \mathsf{hp}_3 \\ \log \mathsf{hp}_4 \\ \log H \end{pmatrix} = \begin{pmatrix} y_1 & 0 & 0 & 0 & 1 \\ 0 & y_2 & 0 & 0 & 1 \\ 0 & 0 & y_1 & 0 & 1 \\ 0 & 0 & 0 & y_2 & 1 \\ \mathsf{sk}r_1y_1 & \mathsf{sk}r_2y_2 & s_1y_1 & s_2y_2 & t+\mathsf{sk}(r_1+r_2) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

The determinant of this matrix is $(y_1.y_2)^2(t-(s_1+s_2)+(\mathsf{sk}(r_1+r_2)-\mathsf{sk}(r_1+r_2))) = (y_1.y_2)^2(t-(s_1+s_2))$, which is non-zero if W does not belong to the language ($t \neq s_1+s_2$). So v is independent from hp and W.▮

Theorem G.5.10 This Smooth Projective Hash Function is pseudo-random under the DLin assumption (the semantic security of the Linear encryption).

Proof: The fact that c_3 really encrypts d_3 that completes well (d_1, d_2) is hidden by the semantic security of the linear encryption, and so under the DLin assumption. So the proof works as above, on the Linear Language. ▮

Combinations. For our blind signature, we want to consider, on input $c = (c_1, c_2, c_3)$ and $b_i = (b_{i,1}, b_{i,2}, b_{i,3})$ for $i = 1, \ldots, \ell$, the language of the $(c, b_1, \ldots, b_\ell)$ such that:

- for each i, $b_i \in \mathsf{BLin}(\mathsf{ek}, u_i) = \mathsf{Lin}(\mathsf{ek}, 1_{\mathbb{G}}) \cup \mathsf{Lin}(\mathsf{ek}, u_i)$;
- if we denote $d_1 = \prod b_{1,i}$ and $d_2 = \prod b_{2,i}$, then we want the plaintext in c to complete (d_1, d_2) into a linear tuple in basis (u, v, vk): $(d_1, d_2, c_1, c_2, c_3) \in \mathsf{ELin}(\mathsf{ek}, \mathsf{vk})$.

This is a conjunction of disjunctions of simple languages: we can use the generic combination [ACP09].

G.6 Security of our Blind Signature

G.6.1 Definition

Definition G.6.1 [Blind Signature Scheme]
A blind signature scheme is defined by three algorithms ($\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSVerif}$) and one interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{U}\rangle$:

- $\mathsf{BSSetup}(1^k)$, generates the global parameters param of the system;
- $\mathsf{BSKeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$: this is an interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, m)$, for a message $m \in \{0,1\}^n$. It generates a signature σ on m under vk related to sk for the user.
- $\mathsf{BSVerif}(\mathsf{vk}, m, \sigma)$ outputs 1 if the signature σ is valid with respect to m and vk, 0 otherwise.

A blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: blindness and unforgeability.

As mentioned above, a blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: blindness and unforgeability.

Blindness states that a malicious signer should be unable to decide which of two messages m_0, m_1 has been signed first in two *valid* executions with an honest user.
Note that the malicious signer $\mathcal{A}$ can choose arbitrarily the keys and thus the verification key vk given to users. However, if $\mathcal{A}$ refuses to sign one of the inputs (i.e. $\sigma_i = \perp$ for $i \in \{0,1\}$) or if one of the signatures is invalid (i.e. $\mathsf{BSVerif}(\mathsf{vk}, m_i, \sigma_i) = 0$ for $i \in \{0,1\}$) then the two resulting signatures are set to $\perp$; the adversary therefore does not gain any advantage if he decides to prevent the normal game execution.

$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk}, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $\sigma_b \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$
4. $\sigma_{1-b} \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, m_{1-b})\rangle$
5. $b^* \leftarrow \mathcal{S}^*(\texttt{GUESS} : \sigma_0, \sigma_1)$;
6. $\texttt{RETURN}\ b^* = b$.

The advantages are

$$\begin{aligned}\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) &= \Pr[\mathsf{Exp}^{\mathsf{bl}-1}_{\mathcal{BS},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{bl}-0}_{\mathcal{BS},\mathcal{A}}(k) = 1] \\ \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS}}(k,t) &= \max_{\mathcal{A} \leq t} \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E},\mathcal{A}}(k).\end{aligned}$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$ for $b \in \{0,1\}$ runs in time at most t. The scheme $\mathcal{BS}$ is deemed blind, if for all polynomials p, $\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E}}(k, p(k))$ is a negligible function of k.

In the security game, we insist on *valid* executions which end with a valid signature σ of the message used by $\mathcal{U}$ under the key vk. The signer could of course send a wrong answer which would lead to an invalid signature. Then, it could easily distinguish a valid signature from an invalid one, and thus the two executions. But this is a kind of denial of service, that is out of scope of this work. This thus means that one valid execution is indistinguishable from other valid executions. This notion was formalized in [HKKL07] and termed *a posteriori blindness*. We enforce this requirement and we add the constraint that even if the signer may deviate arbitrarily from the $\mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$ protocol specification (for $b \in \{0,1\}$) in an attempt to cheat, the signatures σ_0 and σ_1 must be valid with overwhelming probability (i.e. $\mathsf{BSVerif}(\mathsf{vk}, m_0, \sigma_0) = \mathsf{BSVerif}(\mathsf{vk}, m_1, \sigma_1) = 1$ except with negligible probability).

In out model, adversaries are willing to actively cheat but only if they are not caught. It is relevant in contexts where honest behavior cannot be assumed, but where the companies, institutions and individuals involved cannot afford the embarrassment, loss of reputation, and negative press associated with being caught cheating (e.g. e-cash or e-voting). This is similar to the notion of security against covert adversaries from [AL10] and we call this notion *blindness against covert adversaries*.

An adversary against the (one-more) unforgeability tries to generate $q+1$ valid signatures after at most q complete interactions with the honest signer. This security notion can be formalized by the security game $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{U}^*}(k)$ where the adversary is permitted to keep some internal state between the various calls $\texttt{INIT}_i$ (for $i \in \{1,\ldots,q_s\}$), `FIND` and `GUESS`.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k)$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. For $i = 1,\ldots,q_s$, $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{A}(\texttt{INIT}_i : \mathsf{vk})\rangle$
4. $((m_1,\sigma_1),\ldots,(m_{q_s+1},\sigma_{q_s+1})) \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{vk})$;
5. `IF` $\exists i \neq j, m_i = m_j$ `OR` $\exists i, \mathsf{Verif}(\mathsf{pk}, m_i, \sigma_i) = 0$ `RETURN` 0
6. `ELSE RETURN` 1

The success probabilities are

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k) = 1] \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A}\leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$ runs in time at most t. The scheme $\mathcal{S}$ is deemed $\mathsf{EUF}-\mathsf{CMA}$-secure, if for all polynomial p, $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,p(k))$ is a negligible function of k.

Concurrency in the context of blind signatures was put forth by Juels, Luby and Ostrovsky [JLO97] who presented the first security model for blind signatures that takes into account that the adversary may launch many concurrent sessions of the blind signing protocol (operating as either the user or the signer). In this paper, we consider only *round-optimal* blind signatures (i.e. the user sends a single message to the signer and gets a single response) which are concurrently secure.

G.6.2 Security proofs

- $\mathsf{BSSetup}(1^k)$ generates $(p,\mathbb{G},g,\mathbb{G}_T,e)$ and $\mathsf{ek} = (u,v,g) \in \mathbb{G}^3$. It then chooses at random $h \in \mathbb{G}$, $\vec{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}^{\ell+1}$ for the Waters function. It outputs $\mathsf{param} = (p,\mathbb{G},g,\mathbb{G}_T,e,\mathsf{ek},h,\vec{u})$;
- $\mathsf{BSKeyGen}(\mathsf{param})$ picks at $\mathsf{sk} = x$ and computes $\mathsf{vk} = g^x$. vk is public and sk is given to $\mathcal{S}$;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{U}(\mathsf{vk},m)\rangle$: $\mathcal{U}$ wants to get a signature on m
  - $\mathcal{U}$ computes the bit-per-bit encryption of M by encrypting $u_i^{M_i}$ in
  $$b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i}; (r_{i,1}, r_{i,2})),$$
  together with the encryption of $\mathsf{vk}^{r_1+r_2}$ in $c = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}^{r_1+r_2}; (r'_1, r'_2))$, where $r_1 = \sum r_{i,1}$ and $r_2 = \sum r_{i,2}$. $\mathcal{U}$ thus sends
  $$c = (u^{s_1}, v^{s_2}, g^{s_1+s_2}\mathsf{vk}^{r_1+r_2}) \quad b_i = (u^{r_{i,1}}, v^{r_{i,2}}, g^{r_{i,1}+r_{i,2}} u_i^{M_i})$$
  - On input of these ciphertexts, the algorithm $\mathcal{S}$ computes the corresponding SPHF, considering the language $\mathcal{L}$ of valid ciphertexts on an encrypted message. This is the conjunction of several languages:
    1. the one checking that each b_i encrypts a bit;

2. the one checking whether the tuple composed of (d_1, d_2) and the plaintext d_3 in c is a linear tuple in basis (u, v, vk), where $d_1 = \prod_i b_{i,1}$, $d_2 = \prod_i b_{i,2}$, $\delta = u_0 \prod_i b_{i,3}$.

– $\mathcal{S}$ then computes the corresponding Hash-value v, extracts $K = \mathsf{KDF}(v)$, generates $(\sigma_1'' = h^x \delta^s, \sigma_2' = g^s)$ and sends $(\mathsf{hp}, Q = \sigma_1'' \times K, \sigma_2')$;

– Upon receiving $(\mathsf{hp}, Q, \sigma_2')$, using its witnesses and hp, $\mathcal{U}$ computes the $\mathsf{ProjHash}$-value v', extracts $K' = \mathsf{KDF}(v')$ and unmasks $\sigma_2' = Q/K'$. Thanks to the knowledge of r_1 and r_2, it can compute $\sigma_1' = \sigma_1'' \times (\sigma_2')^{-r_1 - r_2}$. Note that if $v' = v$, then $\sigma_1' = h^x \mathcal{F}(M)^s$, which together with $\sigma_2' = g^s$ is a valid Waters signature on M. It can thereafter re-randomize the final signature $\sigma = (\sigma_1' \cdot \mathcal{F}(M)^{s'}, \sigma_2' \cdot g^{s'})$.

- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$, checks whether $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

Proposition G.6.2 This scheme is blind against covert adversaries under the DLin assumption.

$$\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k).$$

Proof: Let us consider an adversary $\mathcal{A}$ against the blindness of our scheme. We build an adversary $\mathcal{B}$ against the DLin assumption.

$\mathcal{G}_0$**:** In a first game $\mathcal{G}_0$, we run the standard protocol:

- $\mathsf{BSSetup}(1^k)$, $\mathcal{B}$ generates $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, $h = g^\alpha$, $\mathsf{ek} = (u, v, g)$ and generators u_i for the Waters function. This constitutes param;
- The adversary $\mathcal{A}$ generates a verification key vk and two messages M^0, M^1.
- $\mathcal{A}$ and $\mathcal{B}$ run twice the interactive issuing protocol, first on the message M^b, and then on the message M^{1-b}:
  – $\mathcal{B}$ generates and sends the $b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i^b}, (r_{i,1}, r_{i,2}))$ and $c = \mathsf{Encrypt}(\mathsf{vk}^{r_1 + r_2})$;
  – $\mathcal{A}$ then outputs $(\mathsf{hp}, Q, \sigma_2')$;
  – $\mathcal{B}$ uses the witnesses and hp to compute v, and so $\sigma_1' = (Q/\mathsf{KDF}(v)) \times {\sigma_2'}^{-r_1 - r_2}$, which together with σ_2' should be a valid Waters Signature on M^b. It then randomizes the signature with s' to get Σ_b.

  The same is done a second time with M^{1-b} to get Σ_{1-b}.
- $\mathcal{B}$ publishes (Σ_0, Σ_1).
- Eventually, $\mathcal{A}$ outputs b'.

We denote by ε the advantage of $\mathcal{A}$ in this game.

$$\begin{aligned} \varepsilon = \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) &= \Pr_{\mathcal{G}_0}[b' = 1 | b = 1] - \Pr_{\mathcal{G}_0}[b' = 1 | b = 0] \\ &= 2 \times \Pr_{\mathcal{G}_0}[b' = b] - 1. \end{aligned}$$

$\mathcal{G}_1$**:** In a second game $\mathcal{G}_1$, we modify the way $\mathcal{B}$ extracts the signatures Σ_b and Σ_{1-b}. One can note that, since we focus one valid executions with the signer, and due to the re-randomization of Waters signatures which leads to random signatures, $\mathcal{B}$ can generates itself random signatures. Knowing α such that $h = g^\alpha$ allows it to compute $\mathsf{sk} = \mathsf{vk}^\alpha$. This game is perfectly indistinguishable from the previous one:

$$\Pr_{\mathcal{G}_1}[b' = b] = \Pr_{\mathcal{G}_0}[b' = b].$$

$\mathcal{G}_2$**:** In the third game, we replace all the ciphertexts sent by $\mathcal{B}$ by encryption of random group elements in $\mathbb{G}$. For proving indistinguishability with the previous game, we use the hybrid technique:

- first, we replace c in the first execution. We then do not need anymore the random coins used in the b_i
- we can now replace one by one the b_i by random encryptions in the first execution
- we then do the same in the second execution

We then use $2 \times (\ell + 1)$ the indistinguishability of the encryption scheme:

$$\varepsilon \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k) + 2 \times \Pr_{\mathcal{G}_2}[b' = b] - 1.$$

In this last game, the two executions are thus perfectly indistinguishable, and thus $\Pr_{\mathcal{G}_2}[b' = b] = 0.5$. ▮

Proposition G.6.3 This scheme is unforgeable under the CDH assumption.

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{A}}(k) \leq \Theta\left(\frac{\mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(k)}{q_s\sqrt{k}}\right)$$

Proof: Let us assume $\mathcal{A}$ is an adversary against the Unforgeability of the scheme. This malicious adversary is able after q_s signing queries to output at least $q_s + 1$ valid signatures on different messages.

We now build an adversary $\mathcal{B}$ against the CDH assumption.

- $\mathcal{B}$ is first given a CDH challenge (g, g^x, h) in a pairing friendly environment $(p, \mathbb{G}, g, \mathbb{G}_T, e)$
- $\mathcal{B}$ emulates BSSetup: it publishes h from its challenge, $\vec{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}^{\ell+1}$ for the Waters function, $\mathsf{ek} = (u = g^a, v = g^b) \in \mathbb{G}^2$, and keeps secret the associated decryption key $\mathsf{dk} = (a, b) \in \mathbb{Z}_p^2$.
- $\mathcal{B}$ then emulates BSKeyGen: it publishes $\mathsf{vk} = g^x$ from the challenge as its verification key (one can note that recovering the signing key h^x is the goal of our adversary $\mathcal{B}$);
- $\mathcal{A}$ can now interact q_s times with the signer, playing the protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{A}\rangle$
  - $\mathcal{A}$ sends the bit-per-bit encryptions b_i, and the extra ciphertext c hiding the verification key raised to the randomness;

– Thanks to dk, $\mathcal{B}$ is able to extract M from the bit-per-bit ciphertexts (either the opening leads to u_i and so $m_i = 1$, or $m_i = 0$), and $Y = \mathsf{vk}^{r_1+r_2}$ from the additional ciphertext c. One can also compute $d_1 = \prod_i b_{i,1} = u^{r_1} = g^{ar_1}$ and $d_2 = \prod_i b_{i,2} = v^{r_2} = g^{br_2}$.

– If one of the extracted terms is not of the right form (either not a bit in the b_i, or $(g, \mathsf{vk}, d_1^{1/a} d_2^{1/b}, Y)$ is not a Diffie-Hellman tuple, which can be checked with a pairing computation), then $\mathcal{A}$ has submitted a "word" not in the appropriate language for the $\mathcal{SPHF}$. Therefore through the smoothness property of the SPHF, it is impossible from a theoretic point of view that the adversary extracts anything from $\mathcal{B}$'s answer, therefore $\mathcal{B}$ simply sends random elements.

– Otherwise, one knows that $d_1^{1/a} d_2^{1/b} = g^{r_1+r_2}$ and $Y = \mathsf{vk}^{r_1+r_2}$.
$\mathcal{B}$ computes $H = -2jq_s + y_0 + \sum y_i M_i$ and $J = z_0 + \sum z_i M_i$, $\mathcal{F}(M) = h^H g^J$. If $H \equiv 0 \mod p$, it aborts, else $\sigma = (\mathsf{vk}^{-J/H} Y^{-1/H} (\mathcal{F}(M) d_1^{1/a} d_2^{1/b})^s, \mathsf{vk}^{-1/H} g^s)$. Defining $t = s - x/H$, we can see this is indeed a valid signature, as we have:

$$\begin{aligned}
\sigma_1 &= \mathsf{vk}^{-J/H} Y^{-1/H} (\mathcal{F}(M) d_1^{1/a} d_2^{1/b})^s = \mathsf{vk}^{-J/H} g^{-x(r_1+r_2)/H} (h^H g^J g^{r_1+r_2})^s \\
&= g^{-xJ/H} g^{-x(r_1+r_2)/H} (h^H g^J g^{r_1+r_2})^t (h^H g^J g^{r_1+r_2})^{x/H} = h^x (h^H g^J g^{r_1+r_2})^t \\
&= \mathsf{sk} \cdot \delta^t \\
\sigma_2 &= \mathsf{vk}^{-1/H} g^s = g^{-x/H} g^s = g^t
\end{aligned}$$

where $\delta = \mathcal{F}(M) \times g^{r_1+r_2}$.

– $\mathcal{B}$ then acts honestly to send the signature through the SPHF.

After a polynomial number of queries $\mathcal{A}$ outputs a valid signature σ^* on a new message M^* with non negligible probability.

- As before $\mathcal{B}$ computes $H^* = -2jq_s + y_0 + \sum y_i M_i^*$ and $J^* = z_0 + \sum z_i M_i^*$, $\mathcal{F}(M) = h^{H^*} g^{J^*}$
- If $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ aborts. Otherwise $\sigma^* = (\mathsf{sk} \cdot \mathcal{F}(M^*)^s, g^s) = (\mathsf{sk} \cdot g^{sJ^*}, g^s)$ and so $\sigma_1^* / {\sigma_2^*}^{J^*} = \mathsf{sk}$. And so $\mathcal{B}$ solves the CDH challenge.

The probability that all the $H \not\equiv 0 \mod p$ for all the simulations, but $H^* \equiv 0 \mod p$ in the forgery is the $(1, q_s)$-programmability of the Waters function. A full proof showing that it happens with probability in $\Theta(\mathsf{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(k)/q_s\sqrt{k})$ can be found in [HK08]. ▌

G.7 Asymmetric Instantiations

G.7.1 Smooth Projective Hash Function

In this subsection, we present the languages we use in our asymmetric instantiations of OSBE and blind signatures.

Diffie Hellman Language. In the following, we will denote $\mathsf{EG}(\mathsf{ek}, M)$ the language of ElGamal encryptions C of the message M under the encryption key $\mathsf{ek} = u$. Clearly, for $M = 1_{\mathbb{G}}$, the language contains the Diffie Hellman pairs in basis (u, g_1). The SPHF system is defined by, for $\mathsf{ek} = u$ and $C = (c_1 = u^r, c_2 = g_1^r \times M)$

$$\begin{array}{ll}
\mathsf{HashKG}(\mathsf{EG}(\mathsf{ek}, M)) = \mathsf{hk} = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2 & \mathsf{Hash}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), C) = c_1^{x_1} (c_2/M)^{x_2} \\
\mathsf{ProjKG}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), C) = \mathsf{hp} = (u^{x_1} g_1^{x_2}) & \mathsf{ProjHash}(\mathsf{hp}, \mathsf{EG}(\mathsf{ek}, M), C, r) = \mathsf{hp}^r
\end{array}$$

Theorem G.7.1 This Smooth Projective Hash Function is correct.

Proof: With the above notations:

- $\mathsf{Hash}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), C) = c_1^{x_1}(c_2/M)^{x_2} = u^{rx_1} g_1^{rx_2}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{EG}(\mathsf{ek}, M), C, r) = \mathsf{hp}^r = (u^{x_1} g_1^{x_2})^r = u^{rx_1} g_1^{rx_2}$

∎

Theorem G.7.2 This Smooth Projective Hash Function is smooth.

Proof: Let us show that from an information theoretic point of view, $v = \mathsf{Hash}(\mathsf{hk}, \mathcal{L}(\mathsf{ek}, M), C)$ is unpredictable, even knowing hp, when C is not a correct ciphertext: $C = (c_1 = u^r, c_2 = g_1^s \times M)$, for $s \neq r$. We recall that $\mathsf{Hash}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), C) = u^{rx_1} g^{sx_2}$ and $\mathsf{hp} = u^{x_1} g_1^{x_2}$: If we denote $u = g_1^y$, we have:

$$\begin{pmatrix} \log \mathsf{hp} \\ \log v \end{pmatrix} = \begin{pmatrix} y & 1 \\ yr & s \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The determinant of this matrix is $y(r-s)$, which is non-zero if C does not belong to the language ($s \neq r$). So v is independent from hp and C.∎

Theorem G.7.3 This Smooth Projective Hash Function is pseudo-random under the DDH assumption in $\mathbb{G}_1$ (the semantic security of the ElGamal encryption).

Proof: As shown above, when c encrypts $M' \neq M$, then the distributions

$$\begin{aligned} \mathcal{D}_1 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \\ \mathcal{D}_2 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, M), c)\} \end{aligned}$$

are perfectly indistinguishable. Under the semantic security of the ElGamal encryption, $\mathcal{E}_{\mathsf{ek}}(M)$ and $\mathcal{E}_{\mathsf{ek}}(M')$ are computationally indistinguishable, and so are the distributions

$$\begin{aligned} \mathcal{D}_0 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \\ \mathcal{D}_1 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v \xleftarrow{\$} \mathbb{G}\} \end{aligned}$$

and the distributions

$$\begin{aligned} \mathcal{D}_2 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M'), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), c)\} \\ \mathcal{D}_3 &= \{\mathsf{EG}(\mathsf{ek}, M), c = \mathcal{E}_{\mathsf{ek}}(M), \mathsf{hp}, v = \mathsf{Hash}(\mathsf{hk}, \mathsf{EG}(\mathsf{ek}, M), c)\} \end{aligned}$$

As a consequence, $\mathcal{D}_0$ and $\mathcal{D}_3$ are computationally indistinguishable, which proves the result. ∎

Bit Encryption Language. In our blind signature protocol, we need to "prove" that a ciphertext encrypts a bit in exponent of a basis u_i. That is the language $\mathsf{BDH}(\mathsf{ek}, u_i) = \mathsf{EG}(\mathsf{ek}, 1_{\mathbb{G}}) \cup \mathsf{EG}(\mathsf{ek}, u_i)$. This is thus a simple disjunction of two $\mathcal{SPHF}$:

- $\mathsf{HashKG}(\mathsf{BDH}(\mathsf{ek}, u_i))$: $\mathsf{hk} = ((x_1, x_2), (y_1, y_2)) \xleftarrow{\$} \mathbb{Z}_p^4$
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{BDH}(\mathsf{ek}, u_i), W)$: $\mathsf{hp} = (u^{x_1} g^{x_2}, u^{y_1} g^{y_2}, \mathsf{hp}_\Delta)$ where
$$\mathsf{hp}_\Delta = c_1^{x_1} c_2^{x_2} \cdot c_1^{y_1} (c_2/u_i)^{y_2}$$
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{BDH}(\mathsf{ek}, u_i), W)$: $v = c_1^{x_1} c_2^{x_2}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{BLin}(\mathsf{ek}, u_i), W, w)$: If $W \in \mathcal{L}_1$, $v' = \mathsf{hp}_1^r$, else (if $W \in \mathcal{L}_2$), $v' = \mathsf{hp}_\Delta / \mathsf{hp}_2^r$

The correctness, smoothness and pseudo-randomness properties of such function directly follow from those of the SPHF on $\mathsf{EG}(\mathsf{pk}, 1_{\mathbb{G}})$ and $\mathsf{EG}(\mathsf{pk}, u_i)$. Each projection key is composed of 3 group elements.

Encrypted Diffie-Hellman Language. We also need to consider a language $\mathsf{EDH}(\mathsf{ek} = u, \mathsf{vk} = (\mathsf{vk}_1 = g_1^x, \mathsf{vk}_2 = g_2^x))$ of tuples (d_1, c_1, c_2), where (c_1, c_2) encrypts d_2 under the public key $\mathsf{ek} = u$, such that (d_1, d_2) is a Diffie Hellman pair in basis (u, vk_1). This can also be expressed as $c_2 = \alpha \times d_2$, where d_2 is the plaintext in (c_1, c_2) under ek, which means that (c_1, α) is a Diffie Hellman pair in basis (u, vk_1), and (d_1, d_2) should be a Diffie Hellman pair in basis (u, vk_1).

More concretely, we consider words $W = (d_1 = u^r, c_1 = u^s, c_2 = g_1^s \cdot \mathsf{vk}_1^r)$, with witness $w = (r, s)$. We have $\alpha = g_1^s$ and $d_2 = \mathsf{vk}_1^r$, but they should remain secret, which requires a specific function, and not a simple conjunction of languages:

- $\mathsf{HashKG}(\mathsf{EDH}(\mathsf{ek}, \mathsf{vk}))$: $\mathsf{hk} = (x_1, x_2, x_3)$
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{EDH}(\mathsf{ek}, \mathsf{vk}), W)$: $\mathsf{hp} = (u^{x_1} g_1^{x_3}, u^{x_2} g_1^{x_3})$
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{EDH}(\mathsf{ek}, \mathsf{vk}), W)$: $v = e(d_1, \mathsf{vk}_2)^{x_1} \cdot e(c_1, g_2)^{x_2} \cdot e(c_2, g_2)^{x_3}$
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{EDH}(\mathsf{ek}, \mathsf{vk}), W, w)$: $v' = e(\mathsf{hp}_1, \mathsf{vk}_2)^r \cdot e(\mathsf{hp}_2, g_2)^s$

We now study the security of our SPHF:

Theorem G.7.4 This Smooth Projective Hash Function is correct.

Proof: With the above notations:

$$\begin{aligned} v &= e(d_1, \mathsf{vk}_2)^{x_1} \cdot e(c_1, g_2)^{x_2} \cdot e(c_2, g_2)^{x_3} = e(u^{xrx_1}, g_2) \cdot e(u^{sx_2}, g_2) \cdot e(g_1^{(xr+s)x_3}, g_2) \\ &= e(u^{xrx_1 + sx_2}, g_2) \cdot e(g_1^{(xr+s)x_3}, g_2) \\ v' &= e(\mathsf{hp}_1, \mathsf{vk}_2)^r \cdot e(\mathsf{hp}_2, g_2)^s = e(u^{xrx_1} g_1^{xrx_3}, g_2) \cdot e(u^{sx_2} g_1^{sx_3}, g_2) \\ &= e(u^{xrx_1 + sx_2}, g_2) \cdot e(g_1^{(xr+s)x_3}, g_2) \end{aligned}$$

∎

Theorem G.7.5 This Smooth Projective Hash Function is smooth.

Proof: Let us show that from an information theoretic point of view, v is unpredictable, even knowing hp, when W is not in the language: $W = (d_1 = u^r, c_1 = u^s, c_2 = g_1^t \cdot \mathsf{vk}_1^r)$, for $t \neq s$. We recall that

$$v = e(u^{xrx_1+sx_2}, g_2) \cdot e(g_1^{(xr+s)x_3}, g_2) = e(H, g)$$

for

$$H = u^{xrx_1+sx_2} \cdot g_1^{(xr+s)x_3}$$

and

$$\mathsf{hp} = (u^{x_1} g_1^{x_3}, u^{x_2} g_1^{x_3})$$

If we denote $u = g_1^y$, we have:

$$\begin{pmatrix} \log \mathsf{hp}_1 \\ \log \mathsf{hp}_2 \\ \log H \end{pmatrix} = \begin{pmatrix} y & 0 & 1 \\ 0 & y & 1 \\ xry & sy & t + xr \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

The determinant of this matrix is $y^2(t - s + (xr - xr)) = y^2(t - s)$, which is non-zero if W does not belong to the language ($t \neq s$). So v is independent from hp and W. ∎

Theorem G.7.6 This Smooth Projective Hash Function is pseudo-random under the DDH assumption (the semantic security of the ElGamal encryption).

Proof: The fact that c_2 really encrypts d_2 that completes well d_1 is hidden by the semantic security of the ElGamal encryption, and so under the DDH assumption. So the proof works as above, on the ElGamal Language. ∎

Combinations. For our blind signature, we want to consider, on input $c = (c_1, c_2)$ and $b_i = (b_{i,1}, b_{i,2})$ for $i = 1, \ldots, \ell$, the language of the $(c, b_1, \ldots, b_\ell)$ such that:

- for each i, $b_i \in \mathsf{BDH}(\mathsf{ek}, u_i) = \mathsf{EG}(\mathsf{ek}, 1_{\mathbb{G}}) \cup \mathsf{EG}(\mathsf{ek}, u_i)$;
- if we denote $d_1 = \prod b_{1,i}$, then we want the plaintext in c to complete d_1 into a linear tuple in basis (u, vk_1): $(d_1, c_1, c_2) \in \mathsf{EDH}(\mathsf{ek}, \mathsf{vk})$.

This is a conjunction of disjunctions of simple languages: we can use the generic combination [ACP09].

G.7.2 OSBE Scheme

Instantiation

We now define our OSBE protocol, where a sender $\mathcal{S}$ wants to send a private message $P \in \{0,1\}^\ell$ to a recipient $\mathcal{R}$ in possession of a Waters signature on a message M.

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter: it first defines an asymmetric pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$, the public parameters $h_1 \xleftarrow{\$} \mathbb{G}_1$ and $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$ for the Waters signature and an encryption key $\mathsf{ek} = g_1^y$, for a random scalar y. All these elements constitute the string param;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$, the authority generates a pair of keys $(\mathsf{sk} = h_1^z, \mathsf{vk} = g_2^z)$ for a random scalar z;

- $\mathsf{OSBESign}(\mathsf{sk}, M)$ produces a signature $\sigma = (h_1^z \mathcal{F}(M)^s, g_1^s, g_2^s)$;
- $\mathsf{OSBEVerif}(\mathsf{vk}, M, \sigma)$ checks if $e(\sigma_1, g_2) = e(\mathcal{F}(M), \sigma_3) \cdot e(h_1, \mathsf{vk})$ and if $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.
- $\mathsf{OSBEProtocol}\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma) \rangle$ runs as follows:
  - $\mathcal{R}$ chooses random $r \xleftarrow{\$} \mathbb{Z}_p$ and sends an ElGamal encryption of σ

    $$C = (c_1 = g_1^r, c_2 = \mathsf{ek}^r \cdot \sigma_1, \sigma_2, \sigma_3)$$

  - $\mathcal{S}$ chooses random $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^3$ and computes:
    - $\mathsf{HashKG}(\mathsf{EG}(\mathsf{ek}, \mathsf{vk}, M)) = \mathsf{hk} = (x_1, x_2)$;
    - $\mathsf{Hash}(\mathsf{hk}; \mathsf{EG}(\mathsf{ek}, \mathsf{vk}, M), C) = v = e(c_1, g_2)^{x_1} (e(c_2, g_2)/(e(h_1, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_3)))^{x_2}$;
    - $\mathsf{ProjKG}(\mathsf{hk}; \mathsf{EG}(\mathsf{ek}, \mathsf{vk}, M), C) = \mathsf{hp} = g_1^{x_1} \mathsf{ek}^{x_2}$;
    - $Q = P \oplus \mathsf{KDF}(v)$.
  - $\mathcal{S}$ then sends (hp, Q) to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = e(\mathsf{hp}^{r_1}, g_2)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

We only use 3 group elements in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$ for the encrypted signature, and we then send back hp, Q. So basically we have 4 elements in $\mathbb{G}_1$, 1 in $\mathbb{G}_2$ and an ℓ-bit string. If we consider standard representation on asymmetric curves, this means the communication costs is approximately of the size of 3 elements on a DLin friendly curve.

Security

To summarize the security of this scheme. This instantiation nearly fits in the high-level instantiation presented before. The difference reside in the part where σ_2, σ_3 are not committed but sent directly. However, due to Waters randomizability, this does not leak any information.

Now, as shown for the high level instantiation, assuming the pseudorandomness of the KDF, the escrow-free property comes from the semantic security of the ElGamal encryption (DDH in $\mathbb{G}_1$), the semantic security comes from both the smoothness of the SPHF (nothing), the unforgeability of Waters signature (CDH^+) and the indistinguishability of the commitment (DDH in $\mathbb{G}_1$), and the semantic security w.r.t. authority comes from the pseudo-randomness of the SPHF (DDH in $\mathbb{G}_1$).

G.7.3 Blind Signature

Let us now present our blind signature, using the above SPHF:

- $\mathsf{BSSetup}(1^k)$, where k is the security parameter, generates a pairing-friendly system

  $$(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$$

  and an ElGamal encryption key $\mathsf{ek} = u \in \mathbb{G}_1$. It also chooses at random $h_1 \in \mathbb{G}_1$ and generators $\vec{u} = (u_i)_{i \in \{0, \ldots, \ell\}} \in \mathbb{G}_1^\ell$ for the Waters function. It outputs the global $\mathsf{param} = (p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, \mathsf{ek}, h_1, \vec{u})$;
- $\mathsf{BSKeyGen}(\mathsf{param})$ picks at random $x \in \mathbb{Z}_p$, sets $\mathsf{sk} = h_1^x$ and computes the verification key $\mathsf{vk} = (g_1^x, g_2^x)$ (note that the two elements, in $\mathbb{G}_1$ and $\mathbb{G}_2$ will be needed);
- $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m) \rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on M

– $\mathcal{U}$ computes the bit-per-bit encryption of M by encrypting $u_i^{M_i}$ in

$$b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i}; r_i),$$

together with the encryption of vk_1^r in $c = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}_1^r; s)$ where $r = \sum r_i$. $\mathcal{U}$ thus sends $c = (u^{s_1}, g_1^s \mathsf{vk}^r)$ and the $b_i = (u^{r_i}, g_1^{r_i} u_i^{M_i})$;

– On input of these ciphertexts, the algorithm $\mathcal{S}$ computes the corresponding $\mathcal{SPHF}$, considering the language $\mathcal{L}$ of valid ciphertexts. This is the conjunction of the several languages presented just before:

1. the one checking that each b_i encrypts a bit: in $\mathsf{BDH}(\mathsf{ek}, u_i)$;
2. the second one considers (d_1, c_1, c_2) and check if (c_1, c_2) encrypts d_2 such that (d_1, d_2) is a Diffie Hellman pair in basis (u, vk_1): in $\mathsf{EDH}(\mathsf{ek}, \mathsf{vk})$ where $d_1 = \prod_i b_{i,1}, \quad \delta = \mathbf{u}_0 \prod_i b_{i,2}$.

Following previous techniques this induces a projection key composed of $3\ell + 2$ elements in $\mathbb{G}_1$.

– $\mathcal{S}$ then computes the corresponding Hash-value v, extracts $K = \mathsf{KDF}(v) \in \mathbb{Z}_p$, generates the blinded signature $(\sigma_1'' = h_1^x \delta^s, \sigma_2' = (g_1^s, g_2^s))$ and sends $(\mathsf{hp}, Q = \sigma_1'' \times g_1^K, \sigma_2')$;

– Upon receiving $(\mathsf{hp}, Q, \sigma_2')$, using its witnesses and hp, $\mathcal{U}$ computes the $\mathsf{ProjHash}$-value v', extracts $K' = \mathsf{KDF}(v')$ and unmasks $\sigma_1'' = Q \times g^{-K'}$. Thanks to the knowledge of r, it can compute $\sigma_1' = \sigma_1'' \times (\sigma_{2,1}')^{-r}$. Note that if $v' = v$, then $\sigma_1' = h_1^x \mathcal{F}(M)^s$, which together with $\sigma_2' = (g_1^s, g_2^s)$ is a valid Waters signature on M. It can thereafter re-randomize the final signature.

- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$, checks whether $e(\sigma_1, g_2) = e(h_1, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2}) \wedge e(\sigma_{2,1}, g_2) = e(g_1, \sigma_{2,2})$.

The whole process requires only $5\ell + 6$ elements in $\mathbb{G}_1$ ($2\ell + 2$ for the ciphertexts, $3\ell + 2$ for the projection key, Q and $\sigma_{2,1}'$) and 1 in $\mathbb{G}_2$ ($\sigma_{2,2}'$), which is way more efficient than the instantiation from [BFPV11] where they required a little more than $6\ell + 7$ group elements in $\mathbb{G}_1$ and $6\ell + 5$ in $\mathbb{G}_2$. Depending on the chosen instantiation for the elliptic curve, elements in $\mathbb{G}_2$ are at least twice bigger than those in $\mathbb{G}_1$ (and even more for higher embedding degree), so our improvement is quite substantial.

The security of this scheme can be proven like the symmetric one, once we have proven the security of the SPHF. One important thing to note, is that it relies on the XDH assumption (DDH is hard in $\mathbb{G}_1$), but not on the SXDH (DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$) as we are used to with Groth-Sahai proofs.

Appendix H

Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages

PKC 2013

[BBC+13a] with F. Benhamouda, O. Blazy, C. Chevalier and D. Pointcheval

Abstract : *Authenticated Key Exchange* (AKE) protocols enable two parties to establish a shared, cryptographically strong key over an insecure network using various authentication means, such as cryptographic keys, short (*i.e.*, low-entropy) secret keys or *credentials*. In this paper, we provide a general framework, that encompasses several previous AKE primitives such as *(Verifier-based) Password-Authenticated Key Exchange* or *Secret Handshakes*, we call *LAKE* for *Language-Authenticated Key Exchange*.
We first model this general primitive in the *Universal Composability* (UC) setting. Thereafter, we show that the Gennaro-Lindell approach can efficiently address this goal. But we need *smooth projective hash functions* on new languages, whose efficient implementations are of independent interest. We indeed provide such hash functions for languages defined by combinations of linear pairing product equations.
Combined with an efficient commitment scheme, that is derived from the highly-efficient UC-secure Lindell's commitment, we obtain a very practical realization of Secret Handshakes, but also *Credential-Authenticated Key Exchange protocols*. All the protocols are UC-secure, in the standard model with a common reference string, under the classical Decisional Linear assumption.

H.1 Introduction

The main goal of an *Authenticated Key Exchange* (AKE) protocol is to enable two parties to establish a shared cryptographically strong key over an insecure network under the complete control of an adversary. AKE is one of the most widely used and fundamental cryptographic primitives. In order for AKE to be possible, the parties must have authentication means, *e.g.* (public or secret) cryptographic keys, short (*i.e.*, low-entropy) secret keys or *credentials* that satisfy a (public or secret) policy.

Motivation.

PAKE, for *Password-Authenticated Key Exchange*, was formalized by Bellovin and Merritt in 1992 [BM92] and followed by many proposals based on different cryptographic assumptions (see [ACP09,CCGS10] and references therein). It allows users to generate a strong cryptographic key based on a shared "human-memorable" (*i.e.* low-entropy) password without requiring a public-key infrastructure. In this setting, an adversary controlling all communication in the network should not be able to mount an off-line dictionary attack.

The concept of *Secret Handshakes* has been introduced in 2003 by Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS+03] (see also [JL09, AKB07]). It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded. In case of failure, the players do not learn any information about the other party's affiliation.

More recently, *Credential-Authenticated Key Exchange* (CAKE) was presented by Camenisch, Casati, Groß and Shoup [CCGS10]. In this primitive, a common key is established if and only if a specific relation is satisfied between credentials hold by the two players. This primitive includes variants of PAKE and Secret Handshakes, and namely Verifier-based PAKE, where the client owns a password pw and the server knows a one-way transformation v of the password only. It prevents massive password recovering in case of server corruption. The two players eventually agree on a common high entropy secret if and only if pw and v match together, and off-line dictionary attacks are prevented for third-party players.

Our Results.

We propose a new primitive that encompasses most of the previous notions of authenticated key exchange. It is closely related to CAKE and we call it LAKE, for *Language-Authenticated Key-Exchange*, since parties establish a common key if and only if they hold credentials that belong to specific (and possibly independent) languages. The definition of the primitive is more practice-oriented than the definition of CAKE from [CCGS10] but the two notions are very similar. In particular, the new primitive enables privacy-preserving authentication and key exchange protocols by allowing two members of the same group to secretly and privately authenticate to each other without revealing this group beforehand.

In order to define the security of this primitive, we use the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provide an ideal functionality for LAKE and give efficient realizations of the new primitive (for a large family of languages) secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions.

We significantly improve the efficiency of several CAKE protocols [CCGS10] for specific languages and we enlarge the set of languages for which we can construct practical schemes. Notably, we obtain a very practical realization of Secret Handshakes and a Verifier-based Password-Authenticated Key Exchange.

Our Techniques.

A general framework to design PAKE in the CRS model was proposed by Gennaro and Lindell [GL03] in 2003. This approach was applied to the UC framework by Canetti, Halevi, Katz, Lindell, and MacKenzie [CHK+05b], and improved by Abdalla, Chevalier and Pointcheval

in [ACP09]. It makes use of the *smooth projective hash functions* (SPHF), introduced by Cramer and Shoup [CS02]. Such a hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projection* key one can only compute the function on a special subset of its domain. Our first contribution is the description of smooth projective hash functions for new interesting languages: Abdalla, Chevalier and Pointcheval [ACP09] explained how to make disjunctions and conjunctions of languages, we study here languages defined by linear pairing product equations on committed values.

In 2011, Lindell [Lin11a] proposed a "highly-efficient" commitment scheme, with a non-interactive opening algorithm, in the UC framework. We will not use it in black-box, but instead we will patch it to make the initial Gennaro and Lindell's approach to work, without zero-knowledge proofs [CHK$^+$05b], using the equivocability of the commitment.

Language Definition.

In [ACP09], Abdalla *et al.* already formalized languages to be considered for SPHF. But, in the following, we will use a more simple formalism, which is nevertheless more general: we consider any efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$, where the additional parameters $\mathsf{pub} \in \{0,1\}^*$ and $\mathsf{priv} \in \mathcal{P}$ define a language $L_{\mathcal{R}}(\mathsf{pub}, \mathsf{priv}) \subseteq \mathcal{S}$ of the words W such that $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1$:

- pub are public parameters;
- priv are private parameters the two players have in mind, and they should think to the same values: they will be committed to, but never revealed;
- W is the word the sender claims to know in the language: it will be committed to, but never revealed.

Our LAKE primitive, specific to two relations $\mathcal{R}_a$ and $\mathcal{R}_b$, will allow two users, Alice and Bob, owning a word $W_a \in L_{\mathcal{R}_a}(\mathsf{pub}, \mathsf{priv}_a)$ and $W_b \in L_{\mathcal{R}_b}(\mathsf{pub}, \mathsf{priv}_b)$ respectively, to agree on a session key under some specific conditions: they first both agree on the public parameter pub, Bob will think about priv'_a for his expected value of priv_a, Alice will do the same with priv'_b for priv_b; eventually, if $\mathsf{priv}'_a = \mathsf{priv}_a$ and $\mathsf{priv}'_b = \mathsf{priv}_b$, and if they both know words in the languages, then the key agreement will succeed. In case of failure, no information should leak about the reason of failure, except the inputs did not satisfy the relations $\mathcal{R}_a$ or $\mathcal{R}_b$, or the languages were not consistent.

We stress that each LAKE protocol will be specific to a pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ describing the way Alice and Bob will authenticate to each other. This pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ specifies the sets $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{S}_a$, $\mathcal{S}_b$ (to which the private parameters and the words should respectively belong). Therefore, the formats of priv_a, priv_b and W_a and W_b are known in advance, but not their values. When $\mathcal{R}_a$ and $\mathcal{R}_b$ are clearly defined from the context (e.g., PAKE), we omit them in the notations. For example, these relations can formalize:

- Password authentication: The language is defined by $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow W = \mathsf{priv}$, and thus $\mathsf{pub} = \emptyset$. The classical setting of PAKE requires the players A and B to use the same password W, and thus we should have $\mathsf{priv}_a = \mathsf{priv}'_b = \mathsf{priv}_b = \mathsf{priv}'_a = W_a = W_b$;
- Signature authentication: $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow \mathsf{Verif}(\mathsf{pub}_1, \mathsf{pub}_2, W) = 1$, where $\mathsf{pub} = (\mathsf{pub}_1 = \mathsf{vk}, \mathsf{pub}_2 = M)$ and $\mathsf{priv} = \emptyset$. The word W is thus a signature of M valid under vk, both specified in pub;

- Credential authentication: we can consider any mix for vk and M in pub or priv, and even in W, for which the relation $\mathcal{R}$ verifies the validity of the signature. When M and vk are in priv or W, we achieve *affiliation-hiding* property.

In the two last cases, the parameter pub can thus consist of a message on which the user is expected to know a signature valid under vk: either the user knows the signing key and can generate the signature on the fly to run the protocol, or the user has been given signatures on some messages (credentials). As a consequence, we just assume that, after having publicly agreed on a common pub, the two players have valid words in the appropriate languages. The way they have obtained these words does not matter.

Following our generic construction, private elements will be committed using encryption schemes, derived from Cramer-Shoup's scheme, and will thus have to be first encoded as n-tuples of elements in a group $\mathbb{G}$. In the case of PAKE, authentication will check that a player knows an appropriate password. The relation is a simple equality test, and accepts for one word only. A random commitment (and thus of a random group element) will succeed with negligible probability. For signature-based authentication, the verification key can be kept secret, but the signature should be unforgeable and thus a random word W should quite unlikely satisfy the relation. We will often make this assumption on useful relations $\mathcal{R}$: for any pub, $\{(\mathsf{priv}, W) \in \mathcal{P} \times \mathcal{S}, \mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1\}$ is sparse (negligible) in $\mathcal{P} \times \mathcal{S}$, and *a fortiori* in the set $\mathbb{G}^n$ in which elements are first embedded.

H.2 Definitions

In this section, we first briefly recall the notations and the security notions of the basic primitives we will use in the rest of the paper, and namely public key encryption and signature. More formal definitions, together with the classical computational assumptions (CDH, DDH, and DLin) are provided in the Appendix H.A.1: A public-key encryption scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $c \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m; r)$, and $m \leftarrow \mathsf{Decrypt}(\mathsf{dk}, c)$. We will need the classical notion of IND-CCA security. A signature scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; s)$, and $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$. We will need the classical notion of EUF-CMA security. In both cases, the global parameters param will be ignored, included in the CRS. We will furthermore make use of collision-resistant hash function families.

H.2.1 Universal Composability

Our main goal will be to provide protocols with security in the universal composability framework. The interested reader is referred to [Can01, CHK+05b] for details. More precisely, we will work in the UC framework with joint state proposed by Canetti and Rabin [CR03] (with the CRS as the joint state). Since players are not individually authenticated, but just afterward if the credentials are mutually consistent with the two players' languages, the adversary will be allowed to interact on behalf of any player from the beginning of the protocol, either with the credentials provided by the environment (static corruption) or without (impersonation attempt). As with the Split Functionality [BCL+05], according to whom sends the first flow for a player, either the player itself or the adversary, we know whether this is an honest player or a dishonest player (corrupted or impersonation attempt, but anyway controlled by the adversary). Then, our goal will be to prove that the best an adversary can do is to try to play against one of the other players, as an honest player would do, with a credential it guessed or obtained in any possible way. This is exactly the so-called one-line dictionary attack when one considers PAKE protocols. In the adaptive corruption setting, the adversary could get complete access to the

private credentials and the internal memory of an honest player, and then get control of it, at any time. But we will restrict to the static corruption setting in this paper. It is enough to deal with most of the concrete requirements: related credentials, arbitrary compositions, and forward-secrecy. To achieve our goal, for a UC-secure LAKE, we will use some other primitives which are secure in the classical setting only.

H.2.2 Commitment

Commitments allow a user to commit to a value, without revealing it, but without the possibility to later change his mind. It is composed of three algorithms: $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k; $\mathsf{Commit}(\ell, m; r)$ produces a commitment c on the input message $m \in \mathcal{M}$ using the random coins $r \xleftarrow{\$} \mathcal{R}$, under the label ℓ, and the opening information d; while $\mathsf{Decommit}(\ell, c, m, d)$ opens the commitment c with the message m and the opening information d that proves the correct opening under the label ℓ.

Such a commitment scheme should be both *hiding*, which says that the commit phase does not leak any information about m, and *binding*, which says that the decommit phase should not be able to open to two different messages. Additional features will be required in the following, such as non-malleability, extractability, and equivocability. We also included a label ℓ, which can be empty or an additional public information that has to be the same in both the commit and the decommit phases. A labelled commitment that is both non-malleable and extractable can be instantiated by an IND-CCA labelled encryption scheme (see the Appendix H.A.1). We will use the Linear Cramer-Shoup encryption scheme [Sha07, CKP07]. We will then patch it, using a technique inspired from [Lin11a], to make it additionally equivocable (see Section H.3). It will have an interactive commit phase, in two rounds: $\mathsf{Commit}(\ell, m; r)$ and a challenge ε from the receiver, which will define an implicit full commitment to be open latter.

H.2.3 Smooth Projective Hash Functions

Smooth projective hash function (SPHF) systems have been defined by Cramer and Shoup [CS02] in order to build a chosen-ciphertext secure encryption scheme. They have thereafter been extended [GL03, ACP09, BPV12b] and applied to several other primitives. Such a system is defined on a language L, with five algorithms:

- $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k;
- $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, L, W)$ derives the projection key hp, possibly depending on a word W;
- $\mathsf{Hash}(\mathsf{hk}, L, W)$ outputs the hash value from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, L, W, w)$ outputs the hash value from the projection key and the witness w that $W \in L$.

The correctness of the scheme assures that if W is in L with w as a witness, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, L, W) = \mathsf{ProjHash}(\mathsf{hp}, L, W, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$. The security is defined through two different notions: the *smoothness* property guarantees that if $W \notin L$, the hash value is *statistically* indistinguishable from a random element, even knowing hp; the *pseudo-randomness* property guarantees that even for a word $W \in L$, but without the knowledge of a witness w, the hash value is *computationally* indistinguishable from a random element, even knowing hp.

H.3 Double Linear Cramer-Shoup Encryption (DLCS)

As explained earlier, any IND-CCA labelled encryption scheme can be used as a non-malleable and extractable labelled commitment scheme: one could use the Cramer-Shoup encryption scheme (see the Appendix H.A.4), but we will focus on the DLin-based primitives, and thus the Linear Cramer-Shoup scheme (see the Appendix H.A.3), we call LCS. Committed/encrypted elements will either directly be group elements, or bit-strings on which we apply a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. In order to add the equivocability, one can use a technique inspired from [Lin11a]. See the Appendix H.B for more details, but we briefly present the commitment scheme we will use in the rest of this paper in conjunction with SPHF.

Linear Cramer-Shoup Commitment Scheme.

The parameters, in the CRS, are a group $\mathbb{G}$ of prime order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$, a collision-resistant hash function $\mathfrak{H}_K$, and possibly an additional reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$ to commit bit-strings. From 9 scalars $(x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, one also sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. The public parameters consist of the encryption key $\mathsf{ek} = (\mathbb{G},\ g_1, g_2, g_3,\ c_1, c_2,\ d_1, d_2,\ h_1, h_2,\ \mathfrak{H}_K)$, while the trapdoor for extraction is $\mathsf{dk} = (x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3)$. One can define the encryption process:

$$\mathsf{LCS}(\ell, \mathsf{ek}, M; r, s) \stackrel{\mathsf{def}}{=} (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^{\xi})^r (c_2 d_2^{\xi})^s)$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$. When ξ is specified from outside, one additionally denotes it

$$\mathsf{LCS}^*(\ell, \mathsf{ek}, M, \xi; r, s).$$

The commitment to a message $M \in \mathbb{G}$, or $M = \mathcal{G}(m)$ for $m \in \{0,1\}^n$, encrypts M under ek: $\mathsf{LCSCom}(\ell, M; r, s) \stackrel{\mathsf{def}}{=} \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s)$. The decommit process consists of M and (r, s) to check the correctness of the encryption. It is possible to do implicit verification, without any decommit information, but just an SPHF on the language of the ciphertexts of M that is privately shared by the two players. Since the underlying encryption scheme is IND-CCA, this commitment scheme is non-malleable and extractable.

Double Linear Cramer-Shoup Commitment Schemes.

To make it equivocable, we double the commitment process, in two steps. The CRS additionally contains a scalar $\aleph \xleftarrow{\$} \mathbb{Z}_p$, one also sets, $\zeta = g_1^{\aleph}$. The trapdoor for equivocability is $\aleph$. The Double Linear Cramer-Shoup encryption scheme, denoted DLCS and detailed in the Appendix H.B is

$$\mathsf{DLCS}(\ell, \mathsf{ek}, M, N; r, s, a, b) \stackrel{\mathsf{def}}{=} (\mathcal{C} \leftarrow \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s), \mathcal{C}' \leftarrow \mathsf{LCS}^*(\ell, \mathsf{ek}, N, \xi; a, b))$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ is computed during the generation of $\mathcal{C}$ and transfered for the generation of $\mathcal{C}'$. As above, we denote DLCSCom denotes the use of DLCS with the encryption key ek. The usual commit/decommit processes are described on Figure H.6 in the Appendix H.B. On Figure H.1, one can find the $\mathsf{DLCSCom}'$ scheme where one can implicitly check the opening with an SPHF. These two constructions essentially differ with $\chi = \mathfrak{H}_K(\mathcal{C}')$ (for the SPHF implicit check) instead of $\chi = \mathfrak{H}_K(M, \mathcal{C}')$ (for the explicit check). We stress that with this alteration, the $\mathsf{DLCSCom}'$ scheme is not a real commitment scheme (not formally extractable/binding): in $\mathsf{DLCSCom}'$, the sender can indeed encrypt M in $\mathcal{C}$ and $N \neq 1_{\mathbb{G}}$ in $\mathcal{C}'$, and then, the global ciphertext $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ contains $M' = MN^{\varepsilon} \neq M$, whereas one would have extracted M from $\mathcal{C}$. But M' is unknown before ε is sent, and thus, if one checks the membership of M' to a sparse language, it will unlikely be true.

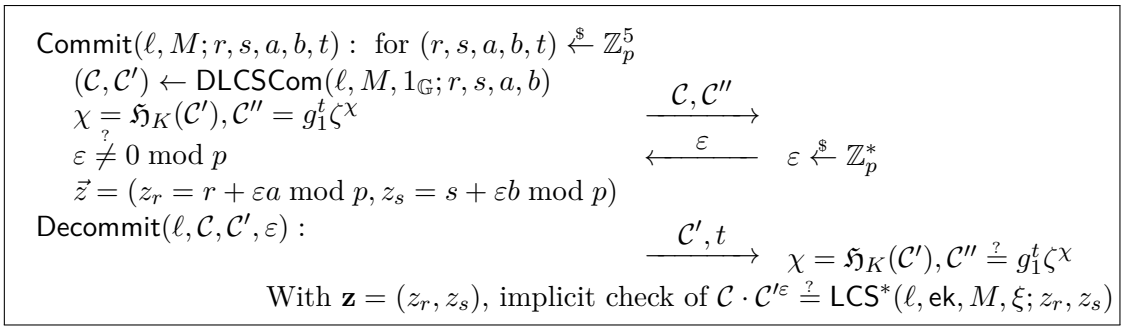


Figure H.1: DLCSCom′ Commitment Scheme for SPHF

Multi-Message Schemes.

One can extend these encryption and commitment schemes to vectors of n messages (see the Appendix H.B). We will denote them n-DLCSCom′ or n-DLCSCom for the commitment schemes. They consist in encrypting each message with independent random coins in $\mathcal{C}_i = (\mathbf{u}_i, e_i, v_i)$ but the same $\xi = \mathfrak{H}_K(\ell, (\mathbf{u}_i), (e_i))$, together with independent companion ciphertexts $\mathcal{C}'_i$ of $1_{\mathbb{G}}$, still with the same ξ for the doubled version. In the latter case, n independent challenges $\varepsilon_i \xleftarrow{\$} \mathbb{Z}_p^*$ are then sent to lead to the full commitment $(\mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon_i}_i)$ with random coins $z_{r_i} = r_i + \varepsilon_i a_i$ and $z_{s_i} = s_i + \varepsilon_i b_i$. Again, if one of the companion ciphertext $\mathcal{C}'_i$ does not encrypt $1_{\mathbb{G}}$, the full commitment encrypts a vector with at least one unpredictable component M'_i. Several non-unity components in the companion ciphertexts would lead to independent components in the full commitment. For languages sparse enough, this definitely turns out not to be in the language.

H.4 SPHF for Implicit Proofs of Membership

In [ACP09], Abdalla *et al.* presented a way to compute a conjunction or a disjunction of languages by some simple operations on their projection keys. Therefore all languages presented afterward can easily be combined together. However as the original set of manageable languages was not really developed, we are going to present several steps to extend it, and namely in order to cover some languages useful in various AKE instantiations.

We will show that almost all the vast family of languages covered by the Groth-Sahai methodology [GS08] can be addressed by our approach too. More precisely, we can handle all the linear pairing product equations, when witnesses are committed using our above (multi-message) DLCSCom′ commitment scheme, or even the non-equivocable LCSCom version. This will be strong enough for our applications. For using them in black-box to build our LAKE protocol, one should note that the projection key is computed from the ciphertext $\mathcal{C}$ when using the simple LCSCom commitment, but also when using the DLCSCom′ version. The full commitment $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ is not required, but ξ only, which is known as soon as $\mathcal{C}$ is given (or the vector $(\mathcal{C}_i)_i$ for the multi-message version). Of course, the hash value will then depend on the full commitment (either $\mathcal{C}$ for the LCSCom commitment, or $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ for the DLCSCom′ commitment).

This will be relevant to our AKE problem: equality of two passwords, in PAKE protocols; corresponding signing/verification keys associated with a valid signature on a pseudonym or a hidden identity, in secret handshakes; valid credentials, in CAKE protocols. All those tests are quite similar: one has to show that the ciphertexts are valid and that the plaintexts satisfy the expected relations in a group. We first illustrate that with commitments of Waters signatures of a public message under a committed verification key. We then explain the general method. The formal proofs are provided in the Appendix H.C.

H.4.1 Commitments of Signatures

Let us consider the Waters signature [Wat05] in a symmetric bilinear group, as reviewed in the Appendix H.A.3, and then we just need to recall that, in a pairing-friendly setting $(p, \mathbb{G}, \mathbb{G}_T, e)$, with public parameters $(\mathcal{F}, g, h)$, and a verification key vk, a signature $\sigma = (\sigma_1, \sigma_2)$ is valid with respect to the message M under the key vk if it satisfies $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

A similar approach has already been followed in [BPV12b], however not with a Linear Cramer-Shoup commitment scheme, nor with such general languages. We indeed first consider the language of the signatures $(\sigma_1, \sigma_2) \in \mathbb{G}^2$ of a message $M \in \{0,1\}^k$ under the verification key $\mathsf{vk} \in \mathbb{G}$, where M is public but vk is private: $L(\mathsf{pub}, \mathsf{priv})$, where $\mathsf{priv} = \mathsf{vk}$ and $\mathsf{pub} = M$. One will thus commit the pair $(\mathsf{vk}, \sigma_1) \in \mathbb{G}^2$ with the label $\ell = (M, \sigma_2)$ using a 2-$\mathsf{DLCSCom}'$ commitment and then prove the commitment actually contains (vk, σ_1) such that $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$. We insist on the fact that σ_1 only has to be encrypted, and not σ_2, in order to hide the signature, since the latter σ_2 is a random group element. If one wants unlinkability between signature commitments, one simply needs to re-randomize (σ_1, σ_2) before encryption. Hence σ_2 can be sent in clear, but bounded to the commitment in the label, together with the pub part of the language. In order to prove the above property on the committed values, we will use conjunctions of SPHF: first, to show that each commitment is well-formed (valid ciphertexts), and then that the associated plaintexts verify the linear pairing equation, where the committed values are underlined: $e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$ Note that vk is not used as a committed value for this verification of the membership of σ to the language since this is the verification key expected by the verifier, specified in the private part priv, which has to be independently checked with respect to the committed verification key. This is enough for the affiliation-hiding property. We could consider the similar language where $M \in \{0,1\}^k$ is in the word too: $e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\underline{\mathcal{F}(M)}, \sigma_2)$, and then one should commit M, bit-by-bit, and then use a $(k+2)$-$\mathsf{DLCSCom}'$ commitment.

H.4.2 Linear Pairing Product Equations

Instead of describing in details the SPHF for the above examples, let us show it for a more general framework: we considered

$$e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2) \text{ or } e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\underline{\mathcal{F}(M)}, \sigma_2),$$

where the unknowns are underlined. These are particular instantiations of t simultaneous equations

$$\Big(\prod_{i \in A_k} e(\underline{\mathcal{Y}_i}, \mathcal{A}_{k,i}) \Big) \cdot \Big(\prod_{i \in B_k} \underline{\mathcal{Z}_i}^{\mathfrak{z}_{k,i}} \Big) = \mathcal{B}_k, \text{ for } k = 1, \ldots, t,$$

where $\mathcal{A}_{k,i} \in \mathbb{G}$, $\mathcal{B}_k \in \mathbb{G}_T$, and $\mathfrak{z}_{k,i} \in \mathbb{Z}_p$, as well as $A_k \subseteq \{1, \ldots, m\}$ and $B_k \subseteq \{m+1, \ldots, n\}$ are public, but the $\mathcal{Y}_i \in \mathbb{G}$ and $\mathcal{Z}_i \in \mathbb{G}_T$ are simultaneously committed using the multi-message $\mathsf{DLCSCom}'$ or LCSCom commitments scheme, in $\mathbb{G}$ or $\mathbb{G}_T$ respectively. This is more general than the relations covered by [CCGS10], since one can also commit scalars bit-by-bit. In the Appendix H.C.4, we detail how to build the corresponding SPHF, and prove the soundness of our approach. For the sake of clarity, we focus here to a single equation only, since multiple equations are just conjunctions. We can even consider the simpler equation $\prod_{i=1}^{i=m} \underline{\mathcal{Z}_i}^{\mathfrak{z}_i} = \mathcal{B}$, since one can lift any ciphertext from $\mathbb{G}$ to a ciphertext in $\mathbb{G}_T$, setting $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$, as well as, for $j = 1, 2, 3$, $G_{i,j} = e(g_j, \mathcal{A}_i)$ and for $j = 1, 2$, $H_{i,j} = e(h_j, \mathcal{A}_i)$, $C_{i,j} = e(c_j, \mathcal{A}_i)$, $D_{i,j} = e(d_j, \mathcal{A}_i)$, to lift all the group basis elements. Then, one transforms $\mathcal{C}_i = \mathsf{LCS}^*(\ell, \mathsf{ek}, \mathcal{Y}_i, \xi; \mathbf{z}_i) = (\vec{u_i} = (g_1^{z_{r_i}}, g_2^{z_{s_i}}, g_3^{z_{r_i}+z_{s_i}}), e_i = h_1^{z_{r_i}} h_2^{z_{s_i}} \cdot \mathcal{Y}_i, v_i = (c_1 d_1^{\xi})^{z_{r_i}} \cdot (c_2 d_2^{\xi})^{z_{s_i}})$ into $(\vec{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^{\xi})^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^{\xi})^{z_{s_i}})$. Encryptions

of $\mathcal{Z}_i$ originally in $\mathbb{G}_T$ use constant basis elements for $j = 1, 2, 3$, $G_{i,j} = G_j = e(g_j, g)$ and for $j = 1, 2$, $H_{i,j} = H_j = e(h_j, g)$, $C_{i,j} = C_j = e(c_j, g)$, $D_{i,j} = D_j = e(d_j, g)$.

The commitments have been generated in $\mathbb{G}$ and $\mathbb{G}_T$ simultaneously using the m-DLCSCom′ version, with a common ξ, where the possible combination with the companion ciphertext to the power ε leads to the above $\mathcal{C}_i$, thereafter lifted to $\mathbb{G}_T$. For the hashing keys, one picks random scalars $(\lambda, (\eta_i, \theta_i, \kappa_i, \mu_i)_{i=1,\ldots,m}) \xleftarrow{\$} \mathbb{Z}_p^{4m+1}$, and sets $\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)$. One then computes the projection keys as $\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2$. The hash value is

$$\prod_i e(u_{i,1}^{\eta_i} \cdot u_{i,2}^{\theta_i} \cdot u_{i,3}^{\kappa_i} \cdot e_i^{\lambda} \cdot v_i^{\mu_i}, \mathcal{A}_i) \cdot \mathcal{B}^{-\lambda} = \prod_i e(\mathsf{hp}_{i,1}^{z_{r_i}} \mathsf{hp}_{i,2}^{z_{s_i}}, \mathcal{A}_i),$$

where $\mathcal{A}_i$ is the constant used to compute $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$ and to lift ciphertexts from $\mathbb{G}$ to $\mathbb{G}_T$, or $\mathcal{A}_i = g^{\mathfrak{z}_i}$ if the ciphertext was already in $\mathbb{G}_T$. These evaluations can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. We insist on the fact that, whereas the hash values are in $\mathbb{G}_T$, the projection keys are in $\mathbb{G}$ even if the ciphertexts are initially in $\mathbb{G}_T$. We stress again that the projection keys require the knowledge of ξ only: known from the LCSCom commitment or the first part $\mathcal{C}$ of the DLCSCom′ commitment.

H.5 Language-Authenticated Key Exchange

H.5.1 The Ideal Functionality

We generalize the Password-Authenticated Key Exchange functionality $\mathcal{F}_{\text{PAKE}}$ (first provided in [CHK+05b]) to more complex languages: the players agree on a common secret key if and only if they own words that lie in the languages the partners have in mind. More precisely, after an agreement on pub between P_i and P_j (modeled here by the use of the split functionality, see below), player P_i uses a word W_i belonging to $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and it expects its partner P_j to use a word W_j belonging to the language $L'_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}'_j)$, and vice-versa for P_j and P_i. We assume relations $\mathcal{R}_i$ and $\mathcal{R}_j$ to be specified by the kind of protocol we study (PAKE, Verifier-based PAKE, secret handshakes, ...) and so the languages are defined by the additional parameters pub, priv_i and priv_j only: they both agree on the public part pub, to be possibly parsed in a different way by each player for each language according to the relations. Note however that the respective languages do not need to be the same or to use similar relations: authentication means could be totally different for the 2 players. The key exchange should succeed if and only if the two following pairs of equations hold: ($L'_i = L_i$ and $W_i \in L_i$) and ($L'_j = L_j$ and $W_j \in L_j$).

Description.

In the initial $\mathcal{F}_{\text{PAKE}}$ functionality [CHK+05b], the adversary was given access to a TestPwd-query, which modeled the on-line dictionary attack. But it is known since [BCL+05] that it is equivalent to use the split functionality model [BCL+05], generate the NewSession-queries corresponding to the corrupted players and tell the adversary (on behalf of the corrupted player) whether the protocol should succeed or not. Both methods enable the adversary to try a credential for a player (on-line dictionary attack). The second method (that we use here) implies allowing $\mathcal{S}$ to ask NewSession-queries on behalf of the corrupted player, and letting it to be aware of the success or failure of the protocol in this case: the adversary learns this information only when it plays on behalf of a player (corruption or impersonation attempt). This is any way an information it would learn at the end of the protocol. We insist that third parties will not learn whether the protocol succeeded or not, as required for secret handshakes. To this aim, the NewKey-query informs in this case the adversary whether the credentials are consistent with the languages or not. In addition, the split functionality model guarantees from the beginning which player is

The functionality $\mathcal{F}_{\text{LAKE}}$ is parametrized by a security parameter k and a public parameter pub for the languages. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- New Session: Upon receiving a query $(\mathsf{NewSession} : \mathsf{sid}, P_i, P_j, W_i, L_i = L(\mathsf{pub}, \mathsf{priv}_i), L'_j = L(\mathsf{pub}, \mathsf{priv}'_j))$ from P_i,
  - If this is the first $\mathsf{NewSession}$-query with identifier sid, record the tuple $(P_i, P_j, W_i, L_i, L'_j, \mathsf{initiator})$. Send $(\mathsf{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ to $\mathcal{S}$ and P_j.
  - If this is the second $\mathsf{NewSession}$-query with identifier sid and there is a record $(P_j, P_i, W_j, L_j, L'_i, \mathsf{initiator})$, record the tuple $(P_j, P_i, W_j, L_j, L'_i, \mathsf{initiator}, W_i, L_i, L'_j, \mathsf{receiver})$. Send $(\mathsf{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{receiver})$ to $\mathcal{S}$ and P_j.
- Key Computation: Upon receiving a query $(\mathsf{NewKey} : \mathsf{sid})$ from $\mathcal{S}$, if there is a record of the form $(P_i, P_j, W_i, L_i, L'_j, \mathsf{initiator}, W_j, L_j, L'_i, \mathsf{receiver})$ and this is the first NewKey-query for session sid, then
  - If $(L'_i = L_i$ and $W_i \in L_i)$ and $(L'_j = L_j$ and $W_j \in L_j)$, then pick a random key sk of length k and store $(\mathsf{sid}, \mathsf{sk})$. In addition, if one player is corrupted, send $(\mathsf{sid}, \mathsf{success})$ to the adversary.
  - Else, store $(\mathsf{sid}, \perp)$, and send $(\mathsf{sid}, \mathsf{fail})$ to the adversary if one player is corrupted.
- Key Delivery: Upon receiving a query $(\mathsf{SendKey} : \mathsf{sid}, P_i, \mathsf{sk})$ from $\mathcal{S}$, then
  - if there is a record of the form $(\mathsf{sid}, \mathsf{sk}')$, then, if both players are uncorrupted, output $(\mathsf{sid}, \mathsf{sk}')$ to P_i. Otherwise, output $(\mathsf{sid}, \mathsf{sk})$ to P_i.
  - if there is a record of the form $(\mathsf{sid}, \perp)$, then pick a random key sk' of length k and output $(\mathsf{sid}, \mathsf{sk}')$ to P_i.

Figure H.2: Ideal Functionality $\mathcal{F}_{\text{LAKE}}$

honest and which one is controlled by the adversary. This finally allows us to get rid of the $\mathsf{TestPwd}$-query. The $\mathcal{F}_{\text{LAKE}}$ functionality is presented in Figure H.2 and the corresponding split functionality $s\mathcal{F}_{\text{LAKE}}$ in Figure H.3, where the languages are formally described and compared using the pub and priv parts.

The security goal is to show that the best attack for the adversary is a basic trial execution with a credential of its guess or choice: the proof will thus consist in emulating any real-life attack by either a trial execution by the adversary, playing as an honest player would do, but with a credential chosen by the adversary or obtained in any way; or a denial of service, where the adversary is clearly aware that its behavior will make the execution fail.

H.5.2 A Generic UC-Secure LAKE Construction

Intuition.

Using smooth projective hash functions on commitments, one can generically define a LAKE protocol as done in [ACP09]. The basic idea is to make the player commit to their private information (for the expected languages and the owned words), and eventually the smooth projective hash functions will be used to make implicit validity checks of the global relation.

Given the functionality $\mathcal{F}_{\text{LAKE}}$, the split functionality $s\mathcal{F}_{\text{LAKE}}$ proceeds as follows:

- Initialization:
  - Upon receiving $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ from party P_i, send $(\mathsf{Init}, \mathsf{sid}, P_i, \mathsf{pub}_i)$ to the adversary.
  - Upon receiving a message $(\mathsf{Init}, \mathsf{sid}, P_i, H, \mathsf{pub}, \mathsf{sid}_H)$ from $\mathcal{S}$, where $H = \{P_i, P_j\}$ is a set of party identities, check that P_i has already sent $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ and that for all recorded $(H', \mathsf{pub}', \mathsf{sid}_{H'})$, either $H = H'$, $\mathsf{pub} = \mathsf{pub}'$ and $\mathsf{sid}_H = \mathsf{sid}_{H'}$ or H and H' are disjoint and $\mathsf{sid}_H \neq \mathsf{sid}_{H'}$. If so, record the pair $(H, \mathsf{pub}, \mathsf{sid}_H)$, send $(\mathsf{Init}, \mathsf{sid}, \mathsf{sid}_H, \mathsf{pub})$ to P_i, and invoke a new functionality $(\mathcal{F}_{\text{LAKE}}, \mathsf{sid}_H, \mathsf{pub})$ denoted as $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ and with set of honest parties H.
- Computation:
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, m)$ from party P_i, find the set H such that $P_i \in H$, the public value pub recorded, and forward m to $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$.
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, P_j, H, m)$ from $\mathcal{S}$, such that $P_j \notin H$, forward m to $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ as if coming from P_j.
  - When $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ generates an output m for party $P_i \in H$, send m to P_i. If the output is for $P_j \notin H$ or for the adversary, send m to the adversary.

Figure H.3: Split Functionality $s\mathcal{F}_{\text{LAKE}}$

To this aim, we use the commitments and associated smooth projective hash functions as described in Sections H.3 and H.4. More precisely, all examples of SPHF in Section H.4 can be used on extractable commitments divided into one or two parts (the non-equivocable LCSCom or the equivocable $\mathsf{DLCSCom}'$ commitments, see Figure H.1). The relations on the committed values will not be explicitly checked, since the values will never be revealed, but will be implicitly checked using SPHF. It is interesting to note that in both cases (one-part or two-part commitment), the projection key will only depend on the first part of the commitment.

As it is often the case in the UC setting, we need the initiator to use stronger primitives than the receiver. They both have to use non-malleable and extractable commitments, but the initiator will use a commitment that is additionally equivocable, the $\mathsf{DLCSCom}'$ in two parts ($(\mathcal{C}_i, \mathcal{C}'_i)$ and $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}_i'^{\vec{\varepsilon}}$), while the receiver will only need the basic LCSCom commitment in one part ($\mathsf{Com}_j = \mathcal{C}_j$).

As already explained, SPHF will be used to implicitly check whether ($L'_i = L_i$ and $W_i \in L_i$) and ($L'_j = L_j$ and $W_j \in L_j$). But since in our instantiations private parameters priv and words W will have to be committed, the structure of these commitments will thus be publicly known in advance: commitments of $\mathcal{P}$-elements and $\mathcal{S}$-elements. Section H.6 discusses on the languages captured by our definition, and illustrates with some AKE protocols. However, while these $\mathcal{P}$ and $\mathcal{S}$ sets are embedded in $\mathbb{G}^n$ from some n, it might be important to prove that the committed values are actually in $\mathcal{P}$ and $\mathcal{S}$ (e.g., one can have to prove it commits bits, whereas messages are first embedded as group elements in $\mathbb{G}$ of large order p). This will be an additional language-membership to prove on the commitments.

This leads to a very simple protocol described on Figure H.4. Note that if a player wants to make external adversaries think he owns an appropriate word, as it is required for Secret Handshakes, he can still play, but will compute everything with dummy words, and will replace the $\mathsf{ProjHash}$ evaluation by a random value, which will lead to a random key at the end.

Security Analysis.

Since we have to assume common pub, we make a first round (with flows in each direction) where the players send their contribution, to come up with pub. These flows will also be used to know if there is a player controlled by the adversary (as with the Split Functionality [BCL$^+$05]). In case the languages have empty pub, these additional flows are not required, since the Split Functionality can be applied on the committed values. The signing key for the receiver is not required anymore since there is one flow only from its side. This LAKE protocol is secure against static corruptions. The proof is provided in the Appendix H.D, and is in the same vein as the one in [CHK$^+$05b, ACP09]. However, it is a bit more intricate:

- in PAKE, when one is simulating a player, and knows the adversary used the correct password, one simply uses this password for the simulated player. In LAKE, when one knows the language expected by the adversary for the simulated player and has to simulate a successful execution (because of success announced by the NewKey-query), one has to actually include a correct word in the commitment: smooth projective hash functions do not allow the simulator to cheat, equivocability of the commitment is the unique trapdoor, but with a valid word. The languages must allow the simulator to produce a valid word W in $L(\mathsf{pub}, \mathsf{priv})$, for any pub and $\mathsf{priv} \in \mathcal{P}$ provided by the adversary or the environment. This will be the case in all the interesting applications of our protocol (see Section H.6): if priv defines a Waters' verification key $\mathsf{vk} = g^x$, with the master key s such that $h = g^s$, the signing key is $\mathsf{sk} = h^x = \mathsf{vk}^s$, and thus the simulator can sign any message; if such a master key does not exist, one can restrict $\mathcal{P}$, and implicitly check it with the SPHF (the additional language-membership check, as said above). But since a random word is generated by the simulator, we need the real player to derive a random word from his own word, and the language to be *self-randomizable*.
- In addition, as already noted, our commitment $\mathsf{DLCSCom}'$ is not formally binding (contrarily to the much less efficient one used in [ACP09]). The adversary can indeed make the extraction give $\vec{M}$ from $\mathcal{C}_i$, whereas Com_i will eventually contain $\vec{M}'$ if $\mathcal{C}'_i$ does not encrypt $(1_{\mathbb{G}})^n$. However, since the actual value $\vec{M}'$ depends on the random challenge $\vec{\varepsilon}$, and the language is assumed sparse (otherwise authentication is easy), the protocol will fail: this can be seen as a denial of service from the adversary.

Theorem H.5.1 Our LAKE scheme from Figure H.4 realizes the $s\mathcal{F}_{\text{LAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$-hybrid model, in the presence of static adversaries, under the DLin assumption and the security of the One-Time Signature.

Actually, from a closer look at the full proof, one can notice that $\mathsf{Com}_j = \mathcal{C}_j$ needs to be extractable, but $\mathsf{IND\text{-}CPA}$ security is enough, which leads to a shorter ciphertext (2 group elements less if one uses a Linear ciphertext instead of LCS). Similarly, one will not have to extract W_i from $\mathcal{C}_i$ when simulating sessions where P_i is corrupted. As a consequence, only the private parts of the languages have to be committed to in Com_i in the first and third rounds, whereas W_i can be encrypted independently with an $\mathsf{IND\text{-}CPA}$ encryption scheme in the third round only (5 group elements less in the first round, and 2 group elements less in the third round if one uses a Linear ciphertext instead of LCS).

H.6 Concrete Instantiations and Comparisons

In this section, we first give some concrete instantiations of several AKE protocols, using our generic protocol of LAKE, and compare the efficiencies of those instantiations.

H.6.1 Possible Languages

As explained above, our LAKE protocol is provably secure for *self-randomizable* languages only. While this notion may seem quite strong, most of the usual languages fall into it. For example, in a PAKE or a Verifier-based PAKE scheme, the languages consist of a single word and so trivially given a word, each user is able to deduce all the words in the language. One may be a little more worried about Waters Signature in our Secret Handshake, and/or Linear pairing equations. However the *self-randomizability* of the languages is easy to show:

- Given a Waters signature $\sigma = (\sigma_1, \sigma_2)$ over a message m valid under a verification key vk, one is able to randomize the signature into any signature over the same message m valid under the same verification key vk simply by picking a random s and computing $\sigma' = (\sigma_1 \cdot \mathcal{F}(m)^s, \sigma_2 \cdot g^s)$.

- For linear pairing equations, with public parameters $\mathcal{A}_i$ for $i = 1, \ldots, m$ and γ_i for $i = m+1, \ldots, n$, and $\mathcal{B}$, given $(\mathcal{X}_1, \ldots, \mathcal{X}_m, \mathcal{Z}_{m+1}, \ldots, \mathcal{Z}_n)$ verifying $\prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{A}_i) \cdot \prod_{i=m+1}^{n} \mathcal{Z}_i^{\gamma_i} = \mathcal{B}$, one can randomize the word in the following way:

  - If $m < n$, one simply picks random $(\mathcal{X}'_1, \ldots, \mathcal{X}'_m)$, $(\mathcal{Z}'_{m+1}, \ldots, \mathcal{Z}'_{n-1})$ and sets $\mathcal{Z}'_n = (\mathcal{B}/(\prod_{i=1}^{m} e(\mathcal{X}'_i, \mathcal{A}_i) \cdot \prod_{i=m+1}^{n-1} \mathcal{Z}_i'^{\gamma_i}))^{1/\gamma_n}$,
  - Else, if $m = n > 1$, one picks random $r_1, \ldots, r_{n-1}$ and sets $\mathcal{X}'_i = \mathcal{X}_i \cdot \mathcal{A}_n^{r_i}$, for $i = 1, \ldots, m-1$ and $\mathcal{X}'_m = \mathcal{X}_m \cdot \prod_{i=1}^{m-1} \mathcal{A}_i^{-r_i}$,
  - Else $m = n = 1$, this means only one word satisfies the equation. So we already have this word.

As we can see most of the common languages manageable with a SPHF are already *self-randomizable*. We now show how to use them in concrete instantiations.

H.6.2 Concrete Instantiations

Password-Authenticated Key Exchange.

Using our generic construction, we can easily obtain a PAKE protocol, as described on Figure H.5, where we optimize from the generic construction, since $\mathsf{pub} = \emptyset$, removing the agreement on pub, but still keeping the one-time signature keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ to avoid man-in-the-middle attacks since it has another later flow: P_i uses a password W_i and expects P_j to own the same word, and thus in the language $L'_j = L_i = \{W_i\}$; P_j uses a password W_j and expects P_i to own the same word, and thus in the language $L'_i = L_j = \{W_j\}$; The relation is the equality test between priv_i and priv_j, which both have no restriction in $\mathbb{G}$ (hence $\mathcal{P} = \mathbb{G}$). As the word W_i, the language private parameters priv_i of a user and priv'_j of the expected language for the other user are the same, each user can commit in the protocol to only one value: its password.

We kept the general description and notations in Figure H.5, but $\mathcal{C}_j$ can be a simply $\mathsf{IND\text{-}CPA}$ encryption scheme. It is quite efficient and relies on the DLin assumption, with DLCS for $(\mathcal{C}_i, \mathcal{C}'_i)$ and thus 10 group elements, but a Linear encryption for $\mathcal{C}_j$ and thus 3 group elements. Projection keys are both 2 group elements. Globally, P_i sends 13 groups elements plus 1 scalar, a verification key and a one-time signature, while P_j sends 5 group elements and 1 scalar: 18 group elements and 2 scalars in total. We can of course instantiate it with the Cramer-Shoup and ElGamal variants, under the DDH assumption: P_i sends 8 groups elements plus 1 scalar, a verification key and a one-time signature, while P_j sends 3 group elements and 1 scalar (all group elements can be in the smallest group): 11 group elements and 2 scalars in total.

Verifier-based PAKE.

The above scheme can be modified into an efficient PAKE protocol that is additionally secure against *server compromise*: the so-called verifier-based PAKE, where the client owns a password pw, while the server knows a verifier only, such as g^{pw}, so that in case of break-in to the server, the adversary will not immediately get all the passwords.

To this aim, as usually done, one first does a PAKE with g^{pw} as common password, then asks the client to additionally prove it can compute the Diffie-Hellman value h^{pw} for a basis h chosen by the server. Ideally, we could implement this trick, where the client P_j just considers the equality test between the g^{pw} and the value committed by the server for the language $L'_i = L_j$, while the server P_i considers the equality test with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, where h is sent as its contribution to the public part of the language by the server $L_i = L'_j$. Since the server chooses h itself, it chooses it as $h = g^{\alpha}$, for an ephemeral random α, and can thus compute $h^{\mathsf{pw}} = (g^{\mathsf{pw}})^{\alpha}$. On its side, the client can compute this value since it knows pw. The client could thus commit to $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, in order to prove its knowledge of pw, whereas the server could just commit to g^{pw}. Unfortunately, from the extractability of the server commitment, one would just get g^{pw}, which is not enough to simulate the client.

To make it in a provable way, the server chooses an ephemeral h as above, and they both run the previous PAKE protocol with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$ as common password, and mutually checked: h is seen as the pub part, hence the preliminary flows are required.

Credential-Authenticated Key Exchange.

In [CCGS10], the authors proposed instantiations of the CAKE primitive for conjunctions of atomic policies that are defined algebraically by relations of the form $\prod_{j=1}^{k} g_j^{F_j} = 1$ where the g_j's are elements of an abelian group and F_j's are integer polynomials in the variables committed by the users.

The core of their constructions relies on their practical UC zero-knowledge proof. There is no precise instantiation of such proof, but it is very likely to be inefficient. Their proof technique indeed requires to transform the underlying Σ-protocols into corresponding Ω-protocols [GMY06] by verifiably encrypting the witness. An Ω-protocol is a Σ-protocol with the additional property that it admits a polynomial-time straight-line extractor. Since the witnesses are scalars in their algebraic relations, their approach requires either inefficient bit-per-bit encryption of these witnesses or Paillier encryption in which case the problem of using group with different orders in the representation and in the encryption requires additional overhead.

Even when used with Σ-protocols, their PAKE scheme without UC-security, requires at least two proofs of knowledge of representations that involve at least 30 group elements (if we assume the encryption to be linear Cramer Shoup), and some extra for the last proof of existence (*cf.* [CKS11]), where our PAKE requires less than 20 group elements. Anyway they say, their PAKE scheme is less efficient than [CHK$^+$05b], which needed 6 rounds and around 30 modular exponentiations per user, while our efficient PAKE requires less than 40 exponentiations, in total, in only 3 rounds. Our scheme is therefore more efficient than the scheme from [CHK$^+$05b] for the same security level (*i.e.* UC-security with static corruptions).

Secret-Handshakes.

We can also instantiate a (linkable) Secret Handshakes protocol, using our scheme with two different languages: P_i will commit to a valid signature σ_i on a message m_i (his identity for example), under a private verification key vk_i, and expects P_j to commit to a valid signature on a message m'_j under a private verification key vk'_j; but P_j will do analogously with a signature σ_j

on m_j under vk_j, while expecting a signature on m'_i under vk'_i. The public parts of the signature (the second component) are sent in clear with the commitments.

In a regular Secret Handshakes both users should use the same languages. But here, we have a more general situation (called *dynamic matching* in [AKB07]): the two participants will have the same final value if and only if they both belong to the organization the other expects. If one lies, our protocol guarantees no information leakage. Furthermore, the semantic security of the session is even guaranteed with respect to the authorities, in a forward-secure way (this property is also achieved in [JL09] but in a weaker security model). Finally, our scheme supports revocation and can handle roles as in [AKB07].

Standard secret handshakes, like [AKB07], usually work with credentials delivered by a unique authority, this would remove our need for a hidden verification key, and private part of the language. Both users would only need to commit to signatures on their identity/credential, and show that they are valid. This would require a dozen of group elements with our approach. Their construction requires only 4 elements under BDH, however it relies on the asymmetric Waters IBE with only two elements, whereas the only security proof known for such IBE [Duc10] requires an extra term in $\mathbb{G}_2$ which would render their technique far less efficient, as several extra terms would be needed to expect a provably secure scheme. While sometimes less effective, our LAKE approach can manage Secret Handshakes, and provide additional functionalities, like more granular control on the credential as part of them can be expressly hidden by both the users. More precisely, we provide affiliation-hiding property and let third parties unaware of the success/failure of the protocol.

Unlinkable Secret-Handshakes.

Moving the users' identity from the public pub part to individual private priv part, and combining our technique with [BPV12b], it is also possible to design an *unlinkable* Secret Handshakes protocol [JL09] with practical efficiency. It illustrates the case where committed values have to be proven in a strict subset of $\mathbb{G}$, as one has to commit to bits: the signed message M is now committed and not in clear, it thus has to be done bit-by-bit since the encoding $\mathcal{G}$ does not allow algebraic operations with the content to apply the Waters function on the message. It is thus possible to prove the knowledge of a Waters signature on a private message (identity) valid under a private verification key. Additional relations can be required on the latter to make authentication even stronger.

Acknowledgments

This work was supported in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet and the ICT Program under Contract ICT-2007-216676 ECRYPT II.

H.A Preliminaries

H.A.1 Formal Definitions of the Primitives

We first recall the definitions of the basic tools, with the security notions with success/advantage that all depend on a security parameter (which is omitted here for simplicity of notation).

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions $\mathfrak{H}_K$ from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^k$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any

adversary $\mathcal{A}$ on a random function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) = \Pr[\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(\mathfrak{H}_K) : \mathfrak{H}_K(m_0) = \mathfrak{H}_K(m_1)],$$
$$\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A})\}.$$

labelled encryption scheme. A labelled public-key encryption scheme is defined by four algorithms:

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key ek and the decryption key dk;
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)) = m.$$

- *Indistinguishability under chosen-ciphertext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls `FIND` and `GUESS`, and makes use of the oracle $\mathsf{ODecrypt}$:

  - $\mathsf{ODecrypt}(\ell, c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key dk. The input queries (ℓ, c) are added to the list $\mathcal{CT}$.

  The advantages are

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A}) = \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(k) = 1]$$
$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A})\}.$$

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\ell, \mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot,\cdot))$
6. `IF` $(\ell^*, c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

labelled commitment scheme. A labelled commitment scheme is defined by three algorithms:

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{Commit}(\ell, m; r)$ produces a commitment c and the opening information d on the input message $m \in \mathcal{M}$ under the label ℓ, using the random coins r;

- $\mathsf{Decommit}(\ell, c, m, d)$ checks the validity of the opening information d on the commitment c for the message m under the label ℓ. It answers 1 for true, and 0 for false.

A commitment scheme $\mathcal{C}$ should satisfy the following properties

- *Correctness*: for any label ℓ, and all messages m, if $(c, d) \leftarrow \mathsf{Commit}(\ell, m; r)$, then $\mathsf{Decommit}(\ell, c, m, d) = 1$.
- *Hiding*: this security notion is similar to the indistinguishability under chosen-plaintext attacks for encryption, which means that c does not help to distinguish between two candidates m_0 and m_1 as committed values.
- *Binding*: this security notion is more an unforgeability notion, which means that for any commitment c, it should be hard to open it in two different ways, which means to exhibit (m_0, d_0) and (m_1, d_1), such that $m_0 \neq m_1$ and

$$\mathsf{Decommit}(\ell, c, m_0, d_0) = \mathsf{Decommit}(\ell, c, m_1, d_1) = 1.$$

The commitment algorithm can be interactive between the sender and the received, but the hiding and the binding properties should still hold. Several additional properties are sometimes required:

- *Extractability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a extractor to get the committed value m from any commitment c. More precisely, if c can be open in a valid way, the extractor can get this value from the commitment.
- *Equivocability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a simulator to generate commitments that can thereafter be open in any way.
- *Non-Malleability*: it should be hard, from a commitment c to generate a new commitment $c' \neq c$ whose committed values are in relation.

It is well-known that any IND-CCA encryption scheme leads to a non-malleable and extractable commitment scheme [GL03].

Signature scheme. A signature scheme is defined by four algorithms:

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the verification key vk and the signing key sk;
- $\mathsf{Sign}(\mathsf{sk}, m; s)$ produces a signature σ on the input message m, under the signing key sk, and using the random coins s;
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

A signature scheme $\mathcal{S}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{vk}, \mathsf{sk})$, all random coins s and all messages m, we have $\mathsf{Verif}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m; s)) = 1$.

- *Existential unforgeability under (adaptive) chosen-message attacks*: this security notion can be formalized by the following security game, where it makes use of the oracle OSign:

  – $\mathsf{OSign}(m)$: This oracle outputs a valid signature on m under the signing key sk. The input queries m are added to the list $\mathcal{SM}$.

  The success probability is:

$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{S},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. `IF` $M \in \mathcal{SM}$ `RETURN` 0
6. `ELSE RETURN` b

$$\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\mathcal{S}}(\mathcal{A}) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k) = 1] \quad \mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\mathcal{S}}(\mathcal{A})\}.$$

Smooth Projective Hash Function. A smooth projective hash function system is defined on a language L, with five algorithms:

- $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k;
- $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, L, W)$ derives the projection key hp, possibly depending on a word W;
- $\mathsf{Hash}(\mathsf{hk}, L, W)$ outputs the hash value from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, L, W, w)$ outputs the hash value from the projection key and the witness w that $W \in L$.

The correctness of the scheme assures that if W is in L with w as a witness, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, L, W) = \mathsf{ProjHash}(\mathsf{hp}, L, W, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$. The security is defined through two different notions, the smoothness and the pseudo-randomness properties, where we use the distribution $\Delta(L, W) = \{(\mathsf{hk}, \mathsf{hp}), \mathsf{hk} \leftarrow \mathsf{HashKG}(L), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, W)\}$:

- the *smoothness* property guarantees that if $W \notin L$, the hash value is *statistically* indistinguishable from a random element, even knowing hp:

  $$\{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \leftarrow \mathsf{Hash}(\mathsf{hk}, L, W)\} \approx_s \{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \xleftarrow{\$} \mathbb{G}\}.$$

  We define by $\mathsf{Adv}^{\mathsf{smooth}}$ the statistical distance between the two distributions.

- the *pseudo-randomness* property guarantees that even for a word $W \in L$, but without the knowledge of a witness w, the hash value is *computationally* indistinguishable from a random element, even knowing hp:

  $$\{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \leftarrow \mathsf{Hash}(\mathsf{hk}, L, W)\} \approx_c \{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \xleftarrow{\$} \mathbb{G}\}.$$

  We define by $\mathsf{Adv}^{\mathsf{pr}}(t)$ the computational distance between the two distributions for t-time distinguishers.

H.A.2 Computational Assumptions

The three classical assumptions we use along this paper are: the computational Diffie-Hellman (CDH), the decisional Diffie-Hellman (DDH) and the decisional Linear (DLin) assumptions. Our constructions essentially rely on the DLin assumption, that implies the CDH. It is the most general since it (presumably) holds in many groups, with or without pairing. Some more efficient instantiations will rely on the DDH assumption but in more specific groups.

Definition H.A.1 [Computational Diffie-Hellman (CDH)] The Computational Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to compute g^{ab}. We define by $\mathsf{Succ}^{\mathsf{cdh}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in finding g^{ab} within time t.

Definition H.A.2 [Decisional Diffie-Hellman (DDH)] The Decisional Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{ddh}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple within time t.

Definition H.A.3 [Decisional Linear Problem (DLin)] The Decisional Linear Problem [BBS04] says that, in a group $(p, \mathbb{G}, g)$, when we are given $(g^x, g^y, g^{xa}, g^{yb}, g^c)$ for unknown random $x, y, a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = a + b \bmod p$ (a linear tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a linear tuple from a random tuple within time t.

H.A.3 Some Primitives in Symmetric Groups – Based on DLin

Linear Cramer-Shoup (LCS) encryption scheme.

The Linear Cramer-Shoup encryption scheme [Sha07] can be tuned to a labelled public-key encryption scheme:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, and sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r, s)$, for a message $M \in \mathbb{G}$ and two random scalars $r, s \xleftarrow{\$} \mathbb{Z}_p$, the ciphertext is $\mathcal{C} = (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^{\xi})^r (c_2 d_2^{\xi})^s)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C} = (\mathbf{u}, e, v))$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \cdot u_3^{x_3+\xi y_3} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^{z_1} u_2^{z_2} u_3^{z_3})$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DLin assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Waters signature.

The Waters signature [Wat05] is defined as follows:

- $\mathsf{Setup}(1^k)$: In a pairing-friendly setting $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, one chooses a random vector $\vec{f} = (f_0, \ldots, f_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ that defines the Waters hash function $\mathcal{F}(M) = f_0 \prod_{i=1}^{k} f_i^{M_i}$ for $M \in \{0,1\}^k$, and an extra generator $h \xleftarrow{\$} \mathbb{G}$. The global parameters param consist of all these elements $(p, \mathbb{G}, g, \mathbb{G}_T, e, \vec{f}, h)$.

- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public verification key as $\mathsf{vk} = g^x$, and the secret signing key as $\mathsf{sk} = h^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g^s)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g) \stackrel{?}{=} e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

This scheme is existentially unforgeable against (adaptive) chosen-message attacks [GMR88] under the CDH assumption.

H.A.4 Some Primitives in Asymmetric Groups – Based on DDH

Cramer-Shoup encryption scheme.

The Cramer-Shoup encryption scheme [CS98] can be tuned into a labelled public-key encryption scheme:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a collision-resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^z)$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Waters signature (asymmetric).

This variant of the Waters signature has been proposed and proved in [BFPV11]:

- $\mathsf{Setup}(1^k)$: In a bilinear group $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e)$, one chooses a random vector $\vec{f} = (f_0, \ldots, f_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, an extra generator $h_1 \xleftarrow{\$} \mathbb{G}_1$. The global parameters param consist of $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e, \vec{f}, h_1)$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public $\mathsf{vk} = \mathfrak{g}_1^x$, and the secret $\mathsf{sk} = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \vec{\sigma_2} = (g_1^s, \mathfrak{g}_1^s))$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether the two equalities $e(\sigma_1, \mathfrak{g}_1) = e(h_1, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_{2,2})$ and $e(\sigma_{2,1}, \mathfrak{g}_1) = e(g_1, \sigma_{2,2})$ hold.

This scheme is unforgeable under the following variant of the CDH assumption:

Definition H.A.4 [The Advanced Computational Diffie-Hellman problem (CDH^+)] In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e)$. The CDH^+ assumption states that given $(g_1, \mathfrak{g}_1, g_1^a, \mathfrak{g}_1^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

H.B Multi Double Linear Cramer-Shoup Commitment

H.B.1 Multi Double Linear Cramer-Shoup ($n-$ DLCS) Encryption

We extend the encryption scheme implicitly presented in Section H.3 to vectors $(M_i, N_i)_{i=1,\ldots,n}$ partially IND-CCA protected with a common ξ. It of course also includes the $n-$ LCS scheme on vectors $(M_i)_i$, when ignoring the $\mathcal{C}'$ part, which is already anyway the case for the decryption oracle:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, and sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. It also chooses a collision-resistant hash function $\mathfrak{H}_K$. The encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, \vec{M}; \vec{r}, \vec{s})$, for a vector $\vec{M} \in \mathbb{G}^n$ and two vectors $\vec{r}, \vec{s} \in \mathbb{Z}_p^n$, computes

  $$\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_n), \text{ where } \mathcal{C}_i = (\mathbf{u}_i = (g_1^{r_i}, g_2^{s_i}, g_3^{r_i+s_i}), e_i = M_i \cdot h_1^{r_i} h_2^{s_i}, v_i = (c_1 d_1^{\xi})^{r_i} (c_2 d_2^{\xi})^{s_i})$$

  with the v_i computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$.
- $\mathsf{Encrypt}'(\ell, \mathsf{ek}, \vec{N}, \xi; \vec{a}, \vec{b})$, for a vector $\vec{N} \in \mathbb{G}^n$ and two vectors $\vec{a}, \vec{b} \in \mathbb{Z}_p^n$, computes

  $$\mathcal{C}' = (\mathcal{C}'_1, \ldots, \mathcal{C}'_n), \text{ where } \mathcal{C}'_i = (\vec{\alpha}_i = (g_1^{a_i}, g_2^{b_i}, g_3^{a_i+b_i}), \beta_i = N_i \cdot h_1^{a_i} h_2^{b_i}, \gamma_i = (c_1 d_1^{\xi})^{a_i} (c_2 d_2^{\xi})^{b_i})$$

  where the γ_i's are computed with the above $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$, hence the additional input.
  One can use both simultaneously: on input $(\ell, \mathsf{ek}, \vec{M}, \vec{N}; \vec{r}, \vec{s}, \vec{a}, \vec{b})$, the global encryption algorithm first calls $\mathsf{Encrypt}(\ell, \mathsf{ek}, \vec{M}; \vec{r}, \vec{s})$ to get $\mathcal{C}$ and ξ and then $\mathsf{Encrypt}'(\ell, \mathsf{ek}, \vec{N}, \xi; \vec{a}, \vec{b})$ to get $\mathcal{C}'$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C}, \mathcal{C}')$: one first parses $\mathcal{C} = (C_1, \ldots, C_n)$ and $\mathcal{C}' = (\mathcal{C}'_1, \ldots, \mathcal{C}'_n)$, where $C_i = (\mathbf{u}_i, e_i, v_i)$ and $\mathcal{C}'_i = (\vec{\alpha}_i, \beta_i, \gamma_i)$, for $i = 1, \ldots, n$, computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$ and checks whether, for $i = 1, \ldots, n$, $u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3} \stackrel{?}{=} v_i$ (but not for the γ_i's). If the equality holds, one computes $M_i = e_i / (u_{i,1}^{z_1} u_{i,2}^{z_2} u_{i,3}^{z_3})$ and $N_i = \beta_i / (\alpha_{i,1}^{z_1} \alpha_{i,2}^{z_2} \alpha_{i,3}^{z_3})$, and outputs $(\vec{M} = (M_1, \ldots, M_n), \vec{N} = (N_1, \ldots, N_n))$. Otherwise, one outputs $\perp$.
- $\mathsf{PDecrypt}(\ell, \mathsf{dk}, \mathcal{C})$: is a partial decryption algorithm that does as above but working on the $\mathcal{C}$ part only to get $\vec{M} = (M_1, \ldots, M_n)$ or $\perp$.

DLCS denotes the particular case where $n = 1$: $\mathsf{DLCS}(\ell, \mathsf{ek}, M, N; r, s, a, b) = (\mathcal{C}, \mathcal{C}')$, with

$$\begin{array}{rcl} \mathcal{C} & = & (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^{\xi})^r (c_2 d_2^{\xi})^s) = \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s), \\ \mathcal{C}' & = & (\vec{\alpha} = (g_1^a, g_2^b, g_3^{a+b}), \beta = N \cdot h_1^a h_2^b, \gamma = (c_1 d_1^{\xi})^a (c_2 d_2^{\xi})^b) = \mathsf{LCS}^*(\ell, \mathsf{ek}, N, \xi; a, b) \end{array}$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.

H.B.2 Security of the Multi Double Linear Cramer Shoup Encryption

Security model.

This scheme is indistinguishable against *partial-decryption chosen-ciphertext* attacks, where a partial-decryption oracle only is available, but even when we allow the adversary to choose $\vec{M}$

and $\vec{N}$ in two different steps (see the security game below), under the DLin assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Indistinguishability against partial-decryption chosen-ciphertext attacks for vectors, in two steps: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls $\texttt{FIND}_M$, $\texttt{FIND}_N$ and $\texttt{GUESS}$. In the first stage $\texttt{FIND}_M$, it receives the encryption key ek; in the second stage $\texttt{FIND}_N$, it receives the encryption of $\vec{M}_b$: $\mathcal{C}^* = \mathsf{Encrypt}(\ell, \mathsf{ek}, \vec{M}_b)$; in the last stage $\texttt{GUESS}$ it receives the encryption of $\vec{N}_b$: $\mathcal{C}'^* = \mathsf{Encrypt}'(\ell, \mathsf{ek}, \xi^*, \vec{N}_b)$, where ξ^* is the value involved in $\mathcal{C}$. During all these stages, it can make use of the oracle $\mathsf{ODecrypt}(\ell, \mathcal{C})$, that outputs the decryption of $\mathcal{C}$ under the label ℓ and the challenge decryption key dk, using $\mathsf{PDecrypt}(\ell, \mathsf{dk}, \mathcal{C})$. The input queries $(\ell, \mathcal{C})$ are added to the list $\mathcal{CT}$.

$\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(k, n)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$; $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
2. $(\ell^*, \vec{M}_0, \vec{M}_1) \leftarrow \mathcal{A}(\texttt{FIND}_M : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
3. $\mathcal{C}^* \leftarrow \mathsf{Encrypt}(\ell^*, \mathsf{ek}, \vec{M}_b)$
4. $(\vec{N}_0, \vec{N}_1) \leftarrow \mathcal{A}(\texttt{FIND}_N : \mathcal{C}^*, \mathsf{ODecrypt}(\cdot,\cdot))$
5. $\mathcal{C}'^* \leftarrow \mathsf{Encrypt}'(\ell^*, \mathsf{ek}, \xi^*, \vec{N}_b)$
6. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \mathcal{C}'^*, \mathsf{ODecrypt}(\cdot,\cdot))$
7. $\texttt{IF}\ (\ell^*, \mathcal{C}^*) \in \mathcal{CT}\ \texttt{RETURN}\ 0$
8. $\texttt{ELSE RETURN}\ b'$

The advantages are, where q_d is the number of decryption queries:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(k,n)=1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}pd\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(k,n)=1]\\ \mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(n, q_d, t) &= \max_{\mathcal{A}\le t} \mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{\mathcal{E}}(\mathcal{A}).\end{aligned}$$

Theorem H.B.1 The Multiple $n-\mathsf{DLCS}$ encryption scheme is IND-PD-CCA if $\mathcal{H}$ is a collision-resistant hash function family, under the DLin assumption in $\mathbb{G}$:

$$\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{n-\mathsf{DLCS}}(n, q_d, t) \le 4n \times \left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) + \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) + \frac{q_d}{p}\right).$$

Corollary H.B.2 The Multiple $n - \mathsf{LCS}$ encryption scheme is IND-CCA if $\mathcal{H}$ is a collision-resistant hash function family, under the DLin assumption in $\mathbb{G}$.

Security proof.

Let us be given a DLin challenge $(g_1, g_2, g_3, u_1 = g_1^r, u_2 = g_2^s, u_3 = g_3^t)$, for which we have to decide whether (u_1, u_2, u_3) is a linear tuple in basis (g_1, g_2, g_3), and thus $t = r + s \bmod p$, or a random one. From an IND-PD-CCA adversary $\mathcal{A}$ against the encryption scheme, we built a DLin distinguisher $\mathcal{B}$. The latter first uses (g_1, g_2, g_3) as the global parameters. It also picks $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \xleftarrow{\$} \mathbb{Z}_p^9$ and sets $c_i = g_i^{x_i} g_3^{x_3}, d_i = g_i^{y_i} g_3^{y_3}, h_i = g_i^{z_i} g_3^{z_3}$, for $i = 1, 2$. It chooses a collision-resistant hash function $\mathfrak{H}_K$ and provides $\mathcal{A}$ with the encryption key $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.

- In the initial game $\mathcal{G}_0$,
  - $\mathcal{A}$'s decryption queries are answered by $\mathcal{B}$, simply using the decryption key dk.

– When $\mathcal{A}$ submits the first challenge vectors

$$\vec{M}_0 = (M_{0,1}, \ldots, M_{0,n}) \text{ and } \vec{M}_1 = (M_{1,1}, \ldots, M_{1,n}),$$

with a label ℓ^*, $\mathcal{B}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$ and encrypts $\vec{M}_b$:

* it chooses two random vectors $\vec{r}^*, \vec{s}^* \xleftarrow{\$} \mathbb{Z}_p^n$
* it defines $\mathcal{C}_i^* = (\mathbf{u}_i^* = (g_1^{r_i^*}, g_2^{s_i^*}, g_3^{r_i^*+s_i^*}), e_i^* = M_{b,i} \cdot h_1^{r_i^*} h_2^{s_i^*}, v_i^* = (c_1 d_1^{\xi^*})^{r_i^*} (c_2 d_2^{\xi^*})^{s_i^*})$, for $i = 1, \ldots, n$, where the v_i^*'s are computed with

$$\xi^* = \mathfrak{H}_K(\ell^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_n^*, e_1^*, \ldots, e_n^*)$$

and $\mathcal{C}^* = (\mathcal{C}_1^*, \ldots, \mathcal{C}_n^*)$.

– When $\mathcal{A}$ submits the second challenge vectors $\vec{N}_0 = (N_{0,1}, \ldots, N_{0,n})$ and $\vec{N}_1 = (N_{1,1}, \ldots, N_{1,n})$,

* $\mathcal{B}$ chooses two random vectors $\vec{a}^*, \vec{b}^* \xleftarrow{\$} \mathbb{Z}_p^n$
* it defines

$$\mathcal{C}_i'^* = (\vec{\alpha}_i^* = (g_1^{a_i^*}, g_2^{b_i^*}, g_3^{a_i^*+b_i^*}), \beta_i^* = N_{b,i} \cdot h_1^{a_i^*} h_2^{b_i^*}, \gamma_i^* = (c_1 d_1^{\xi^*})^{a_i^*} (c_2 d_2^{\xi^*})^{b_i^*})$$

for $i = 1, \ldots, n$, where the γ_i^*'s are computed with the above

$$\xi^* = \mathfrak{H}_K(\ell^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_n^*, e_1^*, \ldots, e_n^*)$$

and $\mathcal{C}'^* = (\mathcal{C}_1'^*, \ldots, \mathcal{C}_n'^*)$.

– When $\mathcal{A}$ returns b', $\mathcal{B}$ outputs $b' \stackrel{?}{=} b$.

$$\Pr_0[1 \leftarrow \mathcal{B}] = \Pr_0[b' = b] = (\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1)/2.$$

- In game $\mathcal{G}_1$, where we assume $t = r + s \bmod p$, to encrypt the challenge vectors $\vec{M}_b$ and $\vec{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}_1^*$: $\mathcal{C}_1^* = (\mathbf{u}_1^* = (u_1, u_2, u_3), e_1^* = M_{b,1} \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}, v_1^* = u_1^{x_1+\xi^* y_1} u_2^{x_2+\xi^* y_2} u_3^{x_3+\xi^* y_3})$, which actually defines $r_1^* = r$ and $s_1^* = s$.

$$\mathbf{u}_1^* = (g_1^{r_1^*}, g_2^{s_1^*}, g_3^{r_1^*+s_1^*}) \qquad e_1^* = M_{b,1} \cdot (g_1^{r_1^*})^{z_1} (g_2^{s_1^*})^{z_2} (g_3^{r_1^*+s_1^*})^{z_3} = M_{b,1} \cdot h_1^{r_1^*} h_2^{s_1^*}$$
$$v_1^* = (g_1^{r_1^*})^{x_1+\xi^* y_1} (g_2^{s_1^*})^{x_2+\xi^* y_2} (g_3^{r_1^*+s_1^*})^{x_3+\xi^* y_3} = (c_1 d_1^{\xi^*})^{r_1^*} (c_2 d_2^{\xi^*})^{s_1^*}$$

The challenge ciphertexts are identical to the encryptions of $\vec{M}_b$ and $\vec{N}_b$ in $\mathcal{G}_0$. Decryption queries are still answered the same way. Hence the gap between this game and the previous game is 0.

$$\Pr_1[1 \leftarrow \mathcal{B}] = \Pr_0[1 \leftarrow \mathcal{B}] = (\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1)/2.$$

- In game $\mathcal{G}_2$, we now assume that $t \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). First, we have to check that the *incorrect* computation of v_1^* does not impact the probability to reject invalid ciphertexts, then we prove that e_1^* is totally independent of $M_{b,1}$.

1. About the validity checks,

$$u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3} \stackrel{?}{=} v_i$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$, three cases can appear with respect to the challenge ciphertext $\mathcal{C}^* = ((\mathbf{u}_1^*, e_1^*, v_1^*), \ldots, (\mathbf{u}_n^*, e_n^*, v_n^*))$:

(a) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) = (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, then necessarily, for some i, $v_i \neq v_i^*$, then the check on index i will fail since one value only is acceptable;

(b) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) \neq (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, but $\xi = \xi^*$, then the adversary has generated a collision for the hash function $\mathfrak{H}_K$.

(c) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) \neq (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, and $\xi \neq \xi^*$: the ciphertext should be accepted iff $v_i = u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3}$, for $i = 1, \ldots, n$. To make it acceptable, if we denote $g_2 = g_1^{\beta_2}$ and $g_3 = g_1^{\beta_3}$, we indeed have

$$\begin{array}{lcccccccc}
\log_{g_1} c_1 & = & x_1 & & +\beta_3 x_3 & & & & \\
\log_{g_1} d_1 & = & & & & & y_1 & & +\beta_3 y_3 \\
\log_{g_1} c_2 & = & & \beta_2 x_2 & +\beta_3 x_3 & & & & \\
\log_{g_1} d_2 & = & & & & & & \beta_3 y_2 & +\beta_3 y_3
\end{array}$$

with in addition,

$$\begin{aligned}
\log_{g_1} v_1^* &= r x_1 + s\beta_2 x_2 + t\beta_3 x_3 + r\xi^* y_1 + s\xi^*\beta_2 y_2 + t\xi^*\beta_3 y_3 \\
\log_{g_1} v_i^* &= r_i^* x_1 + s_i^*\beta_2 x_2 + (r_i^* + s_i^*)\beta_3 x_3 + r_i^*\xi^* y_1 + s_i^*\xi^*\beta_2 y_2 + (r_i^* + s_i^*)\xi^*\beta_3 y_3 \\
&= r_i^* \log_{g_1} c_1 + s_i^* \log_{g_1} c_2 + \xi^* r_i^* \log_{g_1} d_1 + \xi^* s_i^* \log_{g_1} c_2 \\
&\quad (\text{for } i = 2, \ldots, n) \\
\log_{g_1} \gamma_i^* &= a_i^* x_1 + b_i^*\beta_2 x_2 + (a_i^* + b_i^*)\beta_3 x_3 + a_i^*\xi^* y_1 + b_i^*\xi^*\beta_2 y_2 + (a_i^* + b_i^*)\xi^*\beta_3 y_3 \\
&= a_i^* \log_{g_1} c_1 + b_i^* \log_{g_1} c_2 + \xi^* a_i^* \log_{g_1} d_1 + \xi^* b_i^* \log_{g_1} c_2 \\
&\quad (\text{for } i = 1, \ldots, n)
\end{aligned}$$

The $2n-1$ last relations are thus linearly dependent with the 4 above relations, hence remains the useful relations

$$\begin{array}{lcccccccc}
\log_{g_1} c_1 & = & x_1 & & +\beta_3 x_3 & & & & & (1) \\
\log_{g_1} d_1 & = & & & & & y_1 & & +\beta_3 y_3 & (2) \\
\log_{g_1} c_2 & = & & \beta_2 x_2 & +\beta_3 x_3 & & & & & (3) \\
\log_{g_1} d_2 & = & & & & & & \beta_2 y_2 & +\beta_3 y_3 & (4) \\
\log_{g_1} v_1^* & = & r x_1 & +s\beta_2 x_2 & +t\beta_3 x_3 & +r\xi^* y_1 & +s\xi^*\beta_2 y_2 & +t\xi^*\beta_3 y_3 & & (5)
\end{array}$$

One can note that for v_1^* to be predictable, because of the x_1, x_2 and y_1, y_2 components, we need to have (5) $= r$ (1) $+ s$ (3) $+ r\xi^*$ (2) $+ s\xi^*$ (4), and then $t = r + s$, which is not the case, hence v_1^* looks random: in this game, v_1^* is perfectly uniformly distributed in $\mathbb{G}$.
Furthermore, for any v_i in the decryption query, if $\mathbf{u}_i = (g_1^{r'}, g_2^{s'}, g_3^{t'})$ is not a linear triple, then it should be such that

$$\log_{g_1} v_i = r' x_1 + s'\beta_2 x_2 + t'\beta_3 x_3 + r'\xi y_1 + s'\xi\beta_2 y_2 + t'\xi\beta_3 y_3.$$

Since the matrix

$$\begin{pmatrix}
1 & 0 & \beta_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \beta_3 \\
0 & \beta_2 & \beta_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_2 & \beta_3 \\
a & b\beta_2 & c\beta_3 & a\xi^* & b\xi^*\beta_2 & c\xi^*\beta_3 \\
r' & s'\beta_2 & t'\beta_3 & r'\xi & s'\xi\beta_2 & t'\xi\beta_3
\end{pmatrix}$$

has determinant $\beta_2^2\beta_3^2(\xi^* - \xi)(t - r - s)(t' - r' - s') \neq 0$, then the correct value for v_i is unpredictable: an invalid ciphertext will be accepted with probability $1/p$.

2. Let us now consider the mask $u_1^{z_1}u_2^{z_2}u_3^{z_3}$: its discrete logarithm in basis g_1 is $rz_1 + s\beta_2 z_2 + t\beta_3 z_3$, whereas the informations about (z_1, z_2, z_3) are $h_1 = g_1^{z_1}g_3^{z_3}$ and $h_2 = g_2^{z_2}g_3^{z_3}$. The matrix

$$\begin{pmatrix} 1 & 0 & \beta_3 \\ 0 & \beta_2 & \beta_3 \\ r & s\beta_2 & t\beta_3 \end{pmatrix} \quad \text{has determinant } \beta_2\beta_3(t-r-s)(t'-r'-s') \neq 0,$$

then the value of the mask is unpredictable: in this game, e_1^* is perfectly uniformly distributed in $\mathbb{G}$.

Since the unique difference between the two games is the linear/random tuple, unless a collision is found for $\mathfrak{H}_K$ (probability bounded by $\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$) and or an invalid ciphertext is accepted (probability bounded by q_d/p), then

$$\Pr_2[1 \leftarrow \mathcal{B}] \geq \Pr_1[1 \leftarrow \mathcal{B}] - \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}.$$

- In game $\mathcal{G}_3$, to encrypt the challenge vectors $\vec{M}_b$ and $\vec{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}_1^*$: for a random $t_1^* \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{u}_1^* = (g_1^{r_1^*}, g_2^{s_1^*}, g_3^{t_1^*})$, $e_1^* \xleftarrow{\$} \mathbb{G}$, and $v_1^* \xleftarrow{\$} \mathbb{G}$. As just explained, this is perfectly indistinguishable with the previous game:

$$\Pr_3[1 \leftarrow \mathcal{B}] = \Pr_2[1 \leftarrow \mathcal{B}] \geq (\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1)/2 - \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}.$$

- In game $\mathcal{G}_4$, to encrypt the challenge vectors $\vec{M}_b$ and $\vec{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}^*$: for a random vector $\vec{t}^* \xleftarrow{\$} \mathbb{Z}_p^n$, for $i = 2, \ldots, n$: $\mathbf{u}_i^* = (g_1^{r_i^*}, g_2^{s_i^*}, g_3^{t_i^*})$, $e_i^* \xleftarrow{\$} \mathbb{G}$, and $v_i^* \xleftarrow{\$} \mathbb{G}$. Thus replacing sequentially the $\mathcal{C}_i^*$'s by random ones, as we've just done, we obtain

$$\Pr_4[1 \leftarrow \mathcal{B}] \leq \Pr_3[1 \leftarrow \mathcal{B}] - (n-1)\left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right).$$

- In game $\mathcal{G}_5$, to encrypt the challenge vectors $\vec{M}_b$ and $\vec{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}'^*$: for a random vector $\vec{c}^* \xleftarrow{\$} \mathbb{Z}_p^n$, for $i = 1, \ldots, n$: $\vec{\alpha}_i^* = (g_1^{a_i^*}, g_2^{b_i^*}, g_3^{c_i^*})$, $\beta_i^* \xleftarrow{\$} \mathbb{G}$, and $\gamma_i^* \xleftarrow{\$} \mathbb{G}$. Thus replacing sequentially the $\mathcal{C}_i'^*$'s by random ones, as we've just done, we obtain

$$\Pr_5[1 \leftarrow \mathcal{B}] \leq \Pr_4[1 \leftarrow \mathcal{B}] - n\left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right).$$

In this last game, it is clear that $\Pr_5[1 \leftarrow \mathcal{B}] = 1/2$, since $(\vec{M}_b, \vec{N}_b)$ is not used anymore:

$$\frac{\mathsf{Adv}^{\mathsf{ind\text{-}pd\text{-}cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1}{2} - 2n \times \left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right) \leq \frac{1}{2},$$

which concludes the proof.

H.B.3 Double Linear Cramer-Shoup (DLCS) Commitment

Recently, Lindell [Lin11a] proposed a highly efficient UC commitment. Our commitment strongly relies on it, but does not need to be UC secure. We will then show that the decommitment check can be done in an implicit way with an appropriate smooth projective hash function. Basically, the technique consists in encrypting M in $\mathcal{C} = (\mathbf{u}, e, v) = \mathsf{LCS}(\ell, M; r, s)$, also getting

$\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$, and then encrypting $1_\mathbb{G}$ in $\mathcal{C}' = \mathsf{LCS}^*(\ell, 1_\mathbb{G}, \xi; a, b)$, with the same ξ. For a given challenge ε, we can see $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon} = \mathsf{LCS}^*(\ell, M, \xi; r + \varepsilon a, s + \varepsilon b)$, where the computations are done component-wise, as an encryption of M, still using the same above ξ. Note that Lindell [Lin11a] used $\mathcal{C}^{\varepsilon} \cdot \mathcal{C}'$, but our choice seems more natural, since we essentially re-randomize the initial encryption $\mathcal{C}$, but we have to take care of choosing $\varepsilon \neq 0$. It makes use of an equivocable commitment: the Pedersen commitment [Ped91].

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with two independent generators g and ζ;
- $\mathsf{Commit}(m; r)$, for a message $m \xleftarrow{\$} \mathbb{Z}_p$ and random coins $r \xleftarrow{\$} \mathbb{Z}_p$, produces a commitment $c = g^m \zeta^r$;
- $\mathsf{Decommit}(c, m; r)$ outputs m and r, which opens c into m, with checking ability: $c \stackrel{?}{=} g^m \zeta^r$.

This commitment is computationally binding under the discrete logarithm assumption: two different openings (m, r) and (m', r') for a commitment c, leads to the discrete logarithm of ζ in basis g, that is equal to $(m' - m) \cdot (r - r')^{-1} \bmod p$. Granted this logarithm as additional information from the setup, one can equivocate any dummy commitment.

Description.

Our n-message vector commitment, which includes labels, is depicted on Figure H.6, where the computation between vectors are component-wise. We assume we commit vectors of group elements, but they can come from the reversible transformation $\mathcal{G}$. Note that for this commitment scheme, we can use $\vec{\varepsilon} = (\varepsilon, \ldots, \varepsilon)$. For the version with SPHF implicit verification, according to the language, one can have to use independent components $\vec{\varepsilon} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$.

Analysis.

Let us briefly show the properties of this commitment:

- Hiding property: $\vec{M}$ is committed in the Pedersen commitment $\mathcal{C}''$, that does not leak any information, and in the $n - \mathsf{LCS}$ encryption $\mathcal{C}$, that is indistinguishable, even with access to the decryption oracle (extractability). This also implies non-malleability.
- Binding property: $\vec{M}$, after having been hashed, is committed in the Pedersen commitment $\mathcal{C}''$, that is computationally binding.
- Extractability: using the decryption key of the LCS encryption scheme, one can extract $\vec{M}$ from $\mathcal{C}$. Later, one has to open the ciphertext $\mathcal{C}\mathcal{C}'^{\vec{\varepsilon}}$ with $\vec{M}'$, which can be different from $\vec{M}$ in the case that $\mathcal{C}'$ contains $\vec{N} \neq (1_\mathbb{G})^n$. But then $\vec{M}' = \vec{M} \cdot \vec{N}^{\vec{\varepsilon}}$, that is unpredictable at the commit time of $\mathcal{C}''$. With probability at most $1/p$, one can open the commitment with a value $\vec{M}'$ different from $\vec{M}$, if this value $\vec{M}'$ has been correctly anticipated in $\mathcal{C}''$.
- Equivocability: if one wants to open with $\vec{M}'$, one can compute $\vec{N} = (\vec{M}'/\vec{M})^{1/\vec{\varepsilon}}$, encrypt $\vec{N}$ in $\mathcal{C}' = n - \mathsf{LCS}^*(\ell, \vec{N}, \xi; \vec{a}, \vec{b})$, and update χ and t, using the Pedersen trapdoor for equivocability.

To allow an implicit verification with SPHF, one omits to send $\vec{M}$ and $\mathbf{z}$, but make an implicit proof of their existence. Therefore, $\vec{M}$ cannot be committed/verified in $\mathcal{C}''$, which has an impact on the binding property: $\mathcal{C}$ and $\mathcal{C}''$ are not binded to a specific $\vec{M}$, even in a computational way. However, as said above, if $\mathcal{C}''$ contains a ciphertext $\mathcal{C}'$ of $\vec{N} \neq (1_\mathbb{G})^n$, the actual committed value will depend on $\vec{\varepsilon}$: $\vec{M}' = \vec{M}\vec{N}^{\vec{\varepsilon}}$ has its i-component, where $N_i \neq 1_\mathbb{G}$, uniformly distributed in $\mathbb{G}$

when ε is uniformly distributed in $\mathbb{Z}_p^*$. In addition, if $\vec{\varepsilon} \xleftarrow{\$} (\mathbb{Z}_p^*)^n$, all these i-component where $N_i \neq 1_{\mathbb{G}}$ are randomly and independently distributed in $\mathbb{G}$. Then, if the committed value has to satisfy a specific relation, with very few solutions, $\vec{M}'$ will unlikely satisfy it.

H.C Smooth Projective Hash Functions on More Complex Languages

H.C.1 Basic Relations

We first consider Diffie-Hellman pairs and linear tuples and show we can make proof of membership without using any pairing.

DDH pairs.

Let us assume a user is given two elements g, h and then wants to send $G = g^a, H = h^a$ for a chosen a and prove that the pair (G, H) is well-formed with respect to (g, h). We thus consider the language of Diffie Hellman tuples $(g, h, G = g^a, H = h^a)$, with a as a witness.

As done in [CS98], we define a projection key $\mathsf{hp} = g^{x_1}h^{x_2}$ by picking two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret hashing key $\mathsf{hk} = (x_1, x_2)$. One can then compute the hash value in two different ways: $\mathsf{ProjHash}(\mathsf{hp}, (g, h, G, H), a) \stackrel{\mathsf{def}}{=} \mathsf{hp}^a = (g^{ax_1}h^{ax_2}) = G^{x_1}H^{x_2} \stackrel{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, (g, h, G, H))$.

Such SPHF is smooth: this can be seen by proceeding like in the Cramer-Shoup proof. Given $\mathsf{hp} = g^\alpha$, $h = g^\beta$, $G = g^a$ and $H = h^{a'}$, the hash value is g^γ that satisfies:

$$\begin{pmatrix} \alpha \\ \gamma \end{pmatrix} = \begin{pmatrix} 1 & \beta \\ a & \beta a' \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The determinant of this matrix is $\Delta = \beta(a' - a)$, that is zero if and only if we do have a valid Diffie-Hellman tuple. Otherwise, from hp, γ is perfectly hidden, from an information theoretical point of view, and so is $\mathsf{Hash}(\mathsf{hk}, (g, h, G, H))$ too.

DLin tuples.

Let us consider three generators u, v, w, and a tuple $U = u^r, V = v^s, W = w^t$ one wants to prove be linear (*i.e.* $t = r + s$). We first define two projection keys $\mathsf{hp}_1 = u^{x_1}w^{x_3}, \mathsf{hp}_2 = v^{x_2}w^{x_3}$, for random scalars that define the secret hashing key $\mathsf{hk} = (x_1, x_2, x_3)$. One can then compute the hash value in two different ways: $\mathsf{ProjHash}(\mathsf{hp}_1, \mathsf{hp}_2, (u, v, w, U, V, W), r, s) \stackrel{\mathsf{def}}{=} \mathsf{hp}_1^r\mathsf{hp}_2^s = (u^{rx_1}v^{sx_2}w^{x_3(r+s)}) = U^{x_1}V^{x_2}W^{x_3} \stackrel{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, (u, v, w, U, V, W))$.

Once again this SPHF can be shown to be smooth: given $\mathsf{hp}_1 = u^\alpha$, $\mathsf{hp}_2 = u^\beta$, $v = u^\gamma$, $w = u^\delta$, the hash value is u^λ that satisfies:

$$\begin{pmatrix} \alpha \\ \beta \\ \lambda \end{pmatrix} = \begin{pmatrix} 1 & 0 & \delta \\ 0 & \gamma & \delta \\ r & \gamma s & \delta t \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

The determinant of this matrix is $\Delta = \gamma\delta(t - s - r)$, that is zero if and only if we do have a valid linear tuple.

H.C.2 Smooth Projective Hashing on Commitments

We now show that our commitments LCS or DLCS' are well-suited for a use together with smooth projective hash functions: instead of publishing $\vec{z}$ at the decommit phase, in order to check whether $\mathcal{C}\cdot\mathcal{C}'^{\varepsilon} \stackrel{?}{=} \mathsf{LCS}^*(\ell, M, \xi; z_r, z_s)$ (with $\varepsilon = 0$ in the LCS non-equivocable case, or with $\varepsilon \neq 0$ in the DLCS' case), one uses a smooth projective hash function to "implicitly" prove the existence of a witness that the commitment actually contains the claimed (or assumed) value M. We will thereafter be able to use this primitive in Language-Authenticated Key Exchange, for complex languages.

Smooth projective hash functions.

We thus have a commitment, either $\mathcal{C}$ or $\mathcal{C}\cdot\mathcal{C}'^{\varepsilon}$, but we use in both cases the notation $\mathcal{C}$, and want to check whether there exists $\mathbf{z} = (z_r, z_s)$ such that

$$\mathcal{C} = \mathsf{LCS}^*(\ell, M, \xi; z_r, z_s) = (\mathbf{u} = (g_1^{z_r}, g_2^{z_s}, g_3^{z_r+z_s}), e = M\cdot h_1^{z_r}h_2^{z_s}, v = v_1^{z_r}v_2^{z_s}),$$

where we denote $v_1 = c_1 d_1^{\xi}$ and $v_2 = c_2 d_2^{\xi}$. We note here that all the bases g_1, g_2, g_3, h_1, h_2 but also v_1, v_2 are known as soon as ξ is known (the $\mathcal{C}$ part of the DLCS' commitment). One then generates $\mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5$, and derives the projection key that depends on ξ only: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta}g_3^{\kappa}h_1^{\lambda}v_1^{\mu}, \mathsf{hp}_2 = g_2^{\theta}g_3^{\kappa}h_2^{\lambda}v_2^{\mu})$. Then, one can compute the hash value:

$$H = \mathsf{Hash}(\mathsf{hk}, M, \mathcal{C}) \stackrel{\mathsf{def}}{=} u_1^{\eta}u_2^{\theta}u_3^{\kappa}(e/M)^{\lambda}v^{\mu} = \mathsf{hp}_1^{z_r}\mathsf{hp}_2^{z_s} \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, \mathcal{C}; z_r, z_s) = H'.$$

Security properties.

Let us claim and prove the security properties:

Theorem H.C.1 Under the DLin assumption, the above smooth projective hash function is both smooth and pseudo-random:

- Smoothness: $\mathsf{Adv}_{\Pi}^{\mathsf{smooth}} = 0$;
- Pseudo-Randomness: $\mathsf{Adv}_{\Pi}^{\mathsf{pr}}(t) \leq \mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{dlin}}(t)$.

Proof: For the correctness, one can easily check that if $\mathcal{C}$ contains $M = M'$, then $H = H'$:

$$\begin{aligned} H &= u_1^{\eta}u_2^{\theta}u_3^{\kappa}(e/M)^{\lambda}v^{\mu} = (g_1^{z_r})^{\eta}(g_2^{z_s})^{\theta}(g_3^{z_r+z_s})^{\kappa}(h_1^{z_r}h_2^{z_s}M'/M)^{\lambda}(v_1^{z_r}v_2^{z_s})^{\mu} \\ &= (g_1^{\eta}g_3^{\kappa}h_1^{\lambda}v_1^{\mu})^{z_r}\cdot(g_2^{\theta}g_3^{\kappa}h_2^{\lambda}v_2^{\mu})^{z_s}\cdot(M'/M)^{\lambda} = \mathsf{hp}_1^{z_r}\mathsf{hp}_2^{z_s}\cdot(M'/M)^{\lambda} = H'\cdot(M/M')^{\lambda} \end{aligned}$$

Smoothness: if $\mathcal{C}$ is not a correct encryption of M, then H is unpredictable: let us denote M' and z_s' such that $\mathcal{C} = (\vec{u} = (g_1^{z_r}, g_2^{z_s}, g_3^{z_t}), e = M'h_1^{z_r}h_2^{z_s}, v = v_1^{z_r}v_2^{z_s'})$. Then, if we denote $g_2 = g_1^{\beta_2}$ and $g_3 = g_1^{\beta_3}$, and $h_1 = g_1^{\rho_1}$ and $h_2 = g_1^{\rho_2}$, but also $v_1 = g_1^{\delta_1}$ and $v_2 = g_1^{\delta_2}$, and $\Delta = \log_{g_1}(M'/M)$:

$$\begin{aligned} H &= g_1^{\eta z_r}g_1^{\beta_2\theta z_s}g_1^{\beta_3\kappa z_t}(M'/M)^{\lambda}(g_1^{\rho_1 z_r+\rho_2 z_s})^{\lambda}(v_1^{z_r}v_2^{z_s'})^{\mu} \\ \log_{g_1} H &= \eta z_r + \beta_2\theta z_s + \beta_3\kappa z_t + \lambda(\rho_1 z_r + \rho_2 z_s) + \mu(\delta_1 z_r + \delta_2 z_s') + \lambda\Delta \end{aligned}$$

The information leaked by the projected key is $\log_{g_1}\mathsf{hp}_1 = \eta + \beta_3\kappa + \rho_1\lambda + \delta_1\mu$ and $\log_{g_1}\mathsf{hp}_2 = \beta_2\theta + \beta_3\kappa + \rho_2\lambda + \delta_2\mu$, which leads to the matrix

$$\begin{pmatrix} 1 & 0 & \beta_3 & \rho_1 & \delta_1 \\ 0 & \beta_2 & \beta_3 & \rho_2 & \delta_2 \\ z_r & \beta_2 z_s & \beta_3 z_t & \Delta + \rho_1 z_r + \rho_2 z_s & \delta_1 z_r + \delta_2 z_s' \end{pmatrix}$$

One remarks that if $z_t \neq z_r + z_s \bmod p$, then the three rows are not linearly dependent even considering the 3 first components only, and then H is unpredictable. Hence, we can assume that $z_t = z_r + z_s \bmod p$. The third row must thus be the first multiplied by z_r plus the second multiplied by z_s: $\rho_2 z_s = \Delta + \rho_2 z_s \bmod p$ and $z_s = z'_s \bmod p$, which implies $z'_s = s$ and $\Delta = 0$, otherwise, H remains unpredictable.

As a consequence, if $\mathcal{C}$ is not a correct encryption of W, H is perfectly unpredictable in $\mathbb{G}$:

$$\{(\mathsf{hp}, H), \mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5, \mathsf{hp} = (\mathsf{hp}_1 = g_1^\eta g_3^\kappa h_1^\lambda v_1^\mu, \mathsf{hp}_2 = g_2^\theta g_3^\kappa h_2^\lambda v_2^\mu), H \leftarrow \mathsf{Hash}(\mathsf{hk}, M, \mathcal{C})\}$$
$$\approx_s \{(\mathsf{hp}, H), \mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5, \mathsf{hp} = (\mathsf{hp}_1 = g_1^\eta g_3^\kappa h_1^\lambda v_1^\mu, \mathsf{hp}_2 = g_2^\theta g_3^\kappa h_2^\lambda v_2^\mu), H \xleftarrow{\$} \mathbb{G}\}.$$

Pseudo-Randomness: we've just shown that if $\mathcal{C}$ is not a correct encryption of M, then H is statistically unpredictable. Let us be given a triple (g_1, g_2, g_3) together with another triple $\vec{u} = (u_1 = g_1^a, u_2 = g_2^b, u_3 = g_3^c)$. We choose random exponents $(x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$, and for $i = 1, 2$, we set $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. We generate $\mathcal{C} = (\vec{u}, e = M \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}, v = u_1^{x_1+\xi y_1} u_2^{x_2+\xi y_2} u_3^{x_3+\xi y_3})$. If $c = a + b \bmod p$ (i.e., $\vec{u}$ is a linear tuple in basis $\vec{g}$), then $\mathcal{C}$ is a valid encryption of M, otherwise this is not, and we can apply the smoothness property:

$$\mathsf{Adv}^{\mathsf{pr}}_{\Pi}(t) \leq \mathsf{Adv}^{\mathsf{smooth}}_{\Pi} + \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) \leq \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t).$$

∎

H.C.3 Single Equation

Let us assume that we have $\mathcal{Y}_i$ committed in $\mathbb{G}$, in $\vec{c}_i$, for $i = 1, \ldots, m$ and $\mathcal{Z}_i$ committed in $\mathbb{G}_T$, in $\vec{D}_i$, for $i = m+1, \ldots, n$, and we want to show they simultaneously satisfy

$$\left(\prod_{i=1}^{m} e(\mathcal{Y}_i, \mathcal{A}_i)\right) \cdot \left(\prod_{i=m+1}^{n} \mathcal{Z}_i^{\mathfrak{z}_i}\right) = \mathcal{B}$$

where $\mathcal{A}_i \in \mathbb{G}$, $\mathcal{B} \in \mathbb{G}_T$, and $\mathfrak{z}_i \in \mathbb{Z}_p$ are public. As already said, the commitment can either be the LCS or the DLCS' version, but they both come up to a ciphertext $\mathcal{C}$ with the appropriate random coins $\mathbf{z}$:

$$\vec{c}_i = (\vec{u_i} = (g_1^{z_{r_i}}, g_2^{z_{s_i}}, g_3^{z_{r_i}+z_{s_i}}), e_i = h_1^{z_{r_i}} h_2^{z_{s_i}} \cdot \mathcal{Y}_i, v_i = (c_1 d_1^\xi)^{z_{r_i}} \cdot (c_2 d_2^\xi)^{z_{s_i}})$$

for $i = 1, \ldots, m$, which can be transposed into $\mathbb{G}_T$:

$$\vec{C}_i = (\vec{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^\xi)^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^\xi)^{z_{s_i}})$$

for $i = 1, \ldots, m$, where, for $j = 1, 2, 3$, $G_{i,j} = e(g_j, \mathcal{A}_i)$ and for $j = 1, 2$, $H_{i,j} = e(h_j, \mathcal{A}_i)$, $C_{i,j} = e(c_j, \mathcal{A}_i)$, $D_{i,j} = e(d_j, \mathcal{A}_i)$, but also, $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$, and

$$\vec{D}_i = (\vec{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^\xi)^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^\xi)^{z_{s_i}})$$

for $i = m+1, \ldots, n$, where, for $j = 1, 2, 3$, $G_{i,j} = e(g_j, g)$ and for $j = 1, 2$, $H_{i,j} = e(h_j, g)$, $C_{i,j} = e(c_j, g)$, $D_{i,j} = e(d_j, g)$ where g is a generator of $\mathbb{G}$ and

$$\xi = \mathfrak{H}_K(\vec{u_1}, \ldots, \vec{u_m}, \vec{U_{m+1}}, \ldots, \vec{U_n}, e_1, \ldots, e_m, E_{m+1}, \ldots, E_n).$$

$\mathbb{G}$-elements are encrypted under $ek = (\vec{g} = (g_1, g_2, g_3), \vec{h} = (h_1, h_2), \vec{c} = (c_1, d_1), \vec{d} = (c_2, d_2))$, and $\mathbb{G}_T$-element are encrypted under $\mathsf{EK}_i = (\vec{G}_i = (G_{i,1}, G_{i,2}, G_{i,3}), \vec{H}_i = (H_{i,1}, H_{i,2}), \vec{C}_i =$

$(C_{i,1}, C_{i,2}), \vec{D}_i = (D_{i,1}, D_{i,2}))$. Note that an additional label ℓ can be included in the computation of ξ.

For the hashing keys, one picks scalars $(\lambda, (\eta_i, \theta_i, \kappa_i, \mu_i)_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{4n+1}$, and sets $\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)$. One then computes the projection keys as

$$\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2.$$

The associated projection keys in $\mathbb{G}_T$ are $\mathsf{HP}_i = (e(\mathsf{hp}_{i,1}, \mathcal{A}_i), e(\mathsf{hp}_{i,2}, \mathcal{A}_i))$, for $i = 1, \ldots, n$, where $\mathcal{A}_i = g^{\mathfrak{z}_i}$ for $i = m+1, \ldots, n$.

The hash value is

$$\begin{aligned} H &= \left(\prod_i U_{i,1}^{\eta_i} \cdot U_{i,2}^{\theta_i} \cdot U_{i,3}^{\kappa_i} \cdot E_i^{\lambda} \cdot V_i^{\mu_i}\right) \cdot \mathcal{B}^{-\lambda} \\ &= \left(\prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}} \mathcal{Z}_i^{\lambda}\right) \cdot \mathcal{B}^{-\lambda} = \left(\prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}}\right) \left(\left(\prod_{i=1}^{m} e(\mathcal{Y}_i, \mathcal{A}_i)\right) \left(\prod_{i=m+1}^{n} \mathcal{Z}_i\right) / \mathcal{B}\right)^{\lambda} \\ &= \prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}} = \left(\prod_{i=1}^{m} e(\mathsf{hp}_{i,1}^{z_{r_i}}, \mathcal{A}_i) \cdot e(\mathsf{hp}_{i,2}^{z_{s_i}}, \mathcal{A}_i)\right) \cdot \left(e(\prod_{i=m+1}^{n} \mathsf{hp}_{i,1}^{z_{r_i}}, g^{\mathfrak{z}_i}) \cdot e(\prod_{i=m+1}^{n} \mathsf{hp}_{i,2}^{z_{s_i}}, g^{\mathfrak{z}_i})\right) \end{aligned}$$

which can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. We prove below the smoothness, but first extend it even more to several equations.

H.C.4 Multiple Equations

Let us assume that we have $\mathcal{Y}_i$ committed in $\mathbb{G}$, in $\vec{c}_i$, for $i = 1, \ldots, m$ and $\mathcal{Z}_i$ committed in $\mathbb{G}_T$, in $\vec{D}_i$, for $i = m+1, \ldots, n$, and we want to show they simultaneously satisfy

$$\left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}}\right) = \mathcal{B}_k, \text{ for } k = 1, \ldots, t.$$

where $\mathcal{A}_{k,i} \in \mathbb{G}$, $\mathcal{B}_k \in \mathbb{G}_T$, and $\mathfrak{z}_{k,i} \in \mathbb{Z}_p$, as well as $A_k \subseteq \{1, \ldots, m\}$ and $B_k \subseteq \{m+1, \ldots, n\}$ are public. As above, from the commitments, one derives the global ξ, which can also involves the label ℓ, and one can also derive the commitments in $\mathbb{G}_T$, $\vec{C}_{k,i}$ that correspond to the encryption of $\mathcal{Z}_{k,i} = e(\mathcal{Y}_i, \mathcal{A}_{k,i})$ under the keys $\mathsf{EK}_{k,i} = (\vec{G}_{k,i} = (G_{k,i,1}, G_{k,i,2}, G_{k,i,3}), \vec{H}_{k,i} = (H_{k,i,1}, H_{k,i,2}), \vec{C}_{k,i} = (C_{k,i,1}, C_{k,i,2}), \vec{D}_{k,i} = (D_{k,i,1}, D_{k,i,2}))$, where the capital letters $X_{k,i,j}$ correspond to the lower-case letters x_j paired with $\mathcal{A}_{k,i}$.

For the hashing keys, one picks scalars $(\lambda, \{\eta_i, \theta_i, \kappa_i, \mu_i\}_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{4n+1}$, $\{\varepsilon_k\}_{k=1,\ldots,t} \xleftarrow{\$} \mathbb{Z}_p^t$ and sets $\mathsf{hk} = (\{\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)\}_{i=1,\ldots,n}, \{\varepsilon_k\}_{k=1,\ldots,t})$. We insist on the fact that the ε_k's have to be sent after the commitments have been sent, or at least committed to (such as $\mathcal{C}$ and $\mathcal{C}''$ which prevent from any modification). One then computes the projection keys as $\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2$, together with ε_k. The associated projection keys in $\mathbb{G}_T$ are $\mathsf{HP}_{k,i} = (e(\mathsf{hp}_{i,1}, \mathcal{A}_{k,i}), e(\mathsf{hp}_{i,2}, \mathcal{A}_{k,i}))$, for $k = 1, \ldots, t$ and $i = 1, \ldots, n$, where $\mathcal{A}_{k,i} = g^{\mathfrak{z}_{k,i}}$ for $i = m+1, \ldots, n$, together with ε_k. The hash function and the projective hash

function are defined as:

$$\begin{aligned} H &= \prod_k \left(\left(\prod_{i \in A_k \cup B_k} U_{k,i,1}^{\eta_i} \cdot U_{k,i,2}^{\theta_i} \cdot U_{k,i,3}^{\kappa_i} \cdot E_{k,i}^{\lambda} \cdot V_{k,i}^{\mu_i} \right) \cdot \mathcal{B}_k^{-\lambda} \right)^{\varepsilon_k} \\ &= \prod_k \left(\prod_{i \in A_k \cup B_k} \mathsf{HP}_{k,i,1}^{z_{r_i}} \cdot \mathsf{HP}_{k,i,2}^{z_{s_i}} \right)^{\varepsilon_k} \cdot \prod_k \left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{d}_{k,i}} \cdot \mathcal{B}_k^{-1} \right)^{\lambda \varepsilon_k} \\ H' &= \prod_k \left(\prod_{i \in A_k \cup B_k} \mathsf{HP}_{k,i,1}^{z_{r_i}} \cdot \mathsf{HP}_{k,i,2}^{z_{s_i}} \right)^{\varepsilon_k} \end{aligned}$$

which can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. They lead to the same values $H' = H$ if

- for every k, $\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{d}_{k,i}} = \mathcal{B}_k$, which means that all the equations are simultaneously satisfied;
- $\lambda = 0$, which is quite unlikely;
- $\prod_k \Delta_k^{\varepsilon_k} = 1$, where for every k, $\Delta_k = \prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{d}_{k,i}} / \mathcal{B}_k$, which is also quite unlikely since the Δ_k's are fixed before the ε_k's are known.

H.C.5 Security Analysis

Smoothness.

In this section, first we prove the smoothness of the SPHF built right before. For $k = 1$, this proves the smoothness of the SPHF built to handle variables in one linear pairing equation. The list of commitments $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_n)$, which possibly results from the multiplication by the companion ciphertext when using the equivocable variant, should be considered in the language if and only if:

- the commitments are all valid Linear Cramer-Shoup ciphertexts (in either $\mathbb{G}$ or $\mathbb{G}_T$), with the common and fixed ξ;
- the plaintexts satisfy the linear pairing product equations.

Let us assume that one of the commitments is not a valid ciphertext, this means that for some index $i \in \{1, \ldots, n\}$, the ciphertext $(\vec{U}_i = (G_1^{r_i}, G_2^{s_i}, G_3^{t_i}), E_i, V_i)$ in $\mathbb{G}_T$ is such that either $t_i \neq r_i + s_i$ or $V_i \neq (C_1 D_1^{\xi})^{r_i} \cdot (C_2 D_2^{\xi})^{s_i}$. Then, the contribution of this ciphertext in the hash value is $(U_{i,1}^{\eta_i} \cdot U_{i,2}^{\theta_i} \cdot U_{i,3}^{\kappa_i} \cdot E_i^{\lambda} \cdot V_i^{\mu_i})^{\varepsilon'_i}$, where $\varepsilon'_i = \sum_{k, i \in A_k \cup B_k} \varepsilon_k$, knowing the projection keys that reveal, at most,

$$\log_{g_1} \mathsf{hp}_{i,1} = \eta_i + x_3 \cdot \kappa_i + x_4 \cdot \lambda + (y_1 + \xi y_3) \cdot \mu_i \quad \log_{g_1} \mathsf{hp}_{i,2} = x_2 \cdot \theta_i + x_3 \cdot \kappa_i + x_5 \cdot \lambda + (y_2 + \xi y_4) \cdot \mu_i,$$

where $g_2 = g_1^{x_2}$ $g_3 = g_1^{x_3}$ $h_1 = g_1^{x_4}$ $h_2 = g_1^{x_5}$ $c_1 = g_1^{y_1}$ $c_2 = g_1^{y_2}$ $d_1 = g_1^{y_3}$ $d_2 = g_1^{y_4}$. This contribution is thus $(G_1^{r_i \eta_i + x_2 s_i \theta_i + x_3 t_i \kappa_i + z_i \mu_i} \cdot E_i^{\lambda})^{\varepsilon'_i}$, where $V_i = G_1^{z_i}$. But even if all the discrete logarithms were known, and also λ, one has to guess $r_i \eta_i + x_2 s_i \theta_i + x_3 t_i \kappa_i + z_i \mu_i$, given $\eta_i + x_3 \cdot \kappa_i + (y_1 + \xi y_3) \cdot \mu_i$ and $x_2 \cdot \theta_i + x_3 \cdot \kappa_i + (y_2 + \xi y_4) \cdot \mu_i$:

$$\begin{pmatrix} 1 & 0 & x_3 & (y_1 + \xi y_3) \\ 0 & x_2 & x_3 & (y_2 + \xi y_4) \\ r_i & x_2 s_i & x_3 t_i & z_i \end{pmatrix}.$$

The first 3-column matrix has determinant is $x_2x_3(t_i - (r_i + s_i))$, that is non-zero as soon as $t_i \neq r_i + s_i$. In this case, there is no way to guess the correct value better than by chance: $1/p$. If $t_i = (r_i + s_i)$, the third line is linearly dependent with the 2 first, if and only if $z_i = r_i(y_1 + \xi y_3) + s_i(y_2 + \xi y_4)$. Otherwise, one has no better way to guess the value than by chance either. Hence the smoothness of this hash function when one commitment is not valid.

About the equation validity, the E_i's of the involved ciphertexts contain plaintexts $\mathcal{Y}_i$ or $\mathcal{Z}_i$, and contribute to the hash value: from the projection keys, the k-th equation contributes to

$$H_k = \left(\prod_{i\in A_k} \mathsf{HP}_{k,i,1}^{r_i} \cdot \mathsf{HP}_{k,i,2}^{s_i} \cdot \prod_{i\in B_k} \left(\mathsf{HP}_{i,1}^{r_i} \cdot \mathsf{HP}_{i,2}^{s_i}\right)^{\mathfrak{d}_{k,i}}\right)^{\varepsilon_k} \cdot \left(\prod_{i\in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i\in B_k} \mathcal{Z}_i^{\mathfrak{d}_{k,i}} \cdot \mathcal{B}_k^{-1}\right)^{\lambda\varepsilon_k}$$

Let us denote $\alpha_k = \prod_{i\in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i\in B_k} \mathcal{Z}_i^{\mathfrak{d}_{k,i}} \cdot \mathcal{B}_k^{-1}$, then the uncertainty about H is $(\prod_k \alpha_k^{\varepsilon_k})^\lambda$. As soon as one of the equations is not satisfied, one of the α_k is different from 1. Since the ε_k's are unknown at the commitment time, one cannot make the α_k to compensate themselves, but by chance: if one equation is not satisfied, the probability that $\prod_k \alpha_k^{\varepsilon_k} = 1$ is $1/p$. Except this negligible case, $(\prod_k \alpha_k^{\varepsilon_k})^\lambda$ is totally unpredictable since λ is random.

Pseudo-randomness.

The pseudo-randomness can be proven under the DLin assumption: with invalid ciphertexts, the smoothness guarantees unpredictability; without the witnesses, one cannot distinguish a valid ciphertext from an invalid ciphertext.

H.C.6 Asymmetric Setting

Our approach has been presented in the symmetric setting (at least when pairing are required). We can do the same in asymmetric bilinear groups, with $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and even more efficiently, using the Cramer-Shoup encryption scheme, and the analogous n-message commitment scheme, which security relies on the DDH assumption in either $\mathbb{G}_1$ or $\mathbb{G}_2$. In this setting, our methodology can handle linear pairing product equations:

$$\left(\prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i)\right) \cdot \left(\prod_{j=1}^{n} e(\mathcal{A}_j, \mathcal{Y}_j)\right) \cdot \left(\prod_{k=1}^{o} \mathcal{Z}_k^{\mathfrak{d}_k}\right) = g_T,$$

where $\mathcal{A}_j, \mathcal{B}_i, g_T$ are public values, in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively, and $\mathcal{X}_i, \mathcal{Y}_j, \mathcal{Z}_k$ are the unknown values, committed in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively.

H.D Security of the LAKE Protocol: Proof of Theorem H.5.1

For the sake of simplicity, we give in Figure H.7 an explicit version of the protocol described in Figure H.4. We omit the additional verification that all the committed values are in the correct subsets $\mathcal{P}$ and $\mathcal{S}$, since in the proof below we will always easily guarantee this membership. The proof heavily relies on the properties of the commitments and smooth projective hash functions given in Sections H.3, H.4 and Appendix H.B.

H.D.1 Notations

The protocol is played between an initiator, denoted to as P_i, and a receiver, P_j. Each player P_k owns a public part pub_k of a language. Those two public parts pub_i and pub_j will combine to create the common public part pub of the language used in the protocol. Player P_k also owns a

private part priv_k and a word $W_k \in L(\mathsf{pub}, \mathsf{priv}_k)$[1]. It rerandomizes this word W_k into a word V_k still in $L(\mathsf{pub}, \mathsf{priv}_k)$: we assume the languages used to be self-randomizable, which allows such a rerandomization.

We need three different types of commitments for this protocol:

- $\mathsf{EqCommit}$ is an equivocable commitment, such as Pedersen [Ped91], used to engage P_i on its further committed values $\mathcal{C}_i$ and $\mathcal{C}'_i$ with randomness t_i: $\mathcal{C}''_i = \mathsf{EqCommit}((\mathcal{C}_i, \mathcal{C}'_i); t_i)$;
- $\mathsf{EqExtCommit}$ is a labelled equivocable and extractable commitment, used by P_i to commit to its private values (used in the smooth projective hash function) and asking P_j to send a challenge value $\vec{\varepsilon}$.

  It is based on a double encryption scheme (Enc_i and Enc'_i) that is partial-decryption chosen-ciphertext secure (the latter one being strongly related to the former), verifying the following properties, if we denote by $+$ and $\cdot$ two group laws adapted to the schemes:

$$\begin{aligned} \mathcal{C}_i &= \mathsf{Enc}_i(\ell_i, m_i; r_i) \\ \mathcal{C}'_i &= \mathsf{Enc}'_i(\ell_i, n_i; r'_i) \\ \mathsf{Com}_i &= \mathsf{Enc}'_i(\ell_i, m_i \cdot n_i^{\varepsilon}; r_i + \varepsilon r'_i) = \mathcal{C}_i \mathcal{C}_i'^{\varepsilon} \end{aligned}$$

  In the particular cases of (multi) Double-Cramer-Shoup or Double-Linear-Cramer-Shoup, $\mathcal{C}_i$ is a real ciphertext with the correct ξ value, to guarantee non-malleability, but $\mathcal{C}'_i$ and Com_i use the ξ value of $\mathcal{C}_i$. This is the reason why projection keys can be computed as soon as $\mathcal{C}_i$ is known.
- $\mathsf{ExtCommit}$ is a labelled extractable commitment, used by P_j to commit to its private values (used in the smooth projective hash function). It is based on a chosen-ciphertext secure encryption scheme Enc_j which can be equal to Enc_i or different: $\mathsf{Com}_j = \mathcal{C}_j = \mathsf{ExtCommit}(\ell_j, m_j; r_j) = \mathsf{Enc}_j(\ell_j, m_j; r_j)$

Again, note that the projected keys of the smooth projective hash functions depend on $\mathcal{C}_i$ and $\mathcal{C}_j$ only, and do not need Com_i, justifying it can be computed by P_j in $(R2)$, before having actually received $\mathcal{C}'_i$ and thus being able to compute Com_i.

H.D.2 Sketch of Proof

The proof follows that of [CHK+05b] and [ACP09], but with a different approach since we want to prove that the best attack the adversary can perform is to play as an honest player would do with a chosen credential $(\mathsf{pub}_i, \mathsf{priv}_i, \mathsf{priv}'_j, W_i)$ —when trying to impersonate P_i— or $(\mathsf{pub}_j, \mathsf{priv}_j, \mathsf{priv}'_i, W_j)$ —when trying to impersonate P_j—. In order to prove Theorem H.5.1, we need to construct, for any real-world adversary $\mathcal{A}$ (controlling some dishonest parties), an ideal-world adversary $\mathcal{S}$ (interacting with dummy parties and the split functionality $s\mathcal{F}_{\text{LAKE}}$) such that no environment $\mathcal{Z}$ can distinguish between an execution with $\mathcal{A}$ in the real world and $\mathcal{S}$ in the ideal world with non-negligible probability.

The split functionality $s\mathcal{F}_{\text{LAKE}}$ is defined in Section H.5, following [BCL+05]. In particular, we assume that at the beginning of the protocol, $\mathcal{S}$ receives from it the contribution pub_i of P_i to the public language pub as answer to the Init query sent by the environment on behalf of this player. The preflow phase will determine the whole public language pub.

[1] Since pub is unknown before the beginning of the protocol, one can imagine that P_k knows several words W_k, corresponding to different possibilities for the public part pub_ℓ its partner can choose. Once pub is set, P_k chooses a word $W_k \in L(\mathsf{pub}, \mathsf{priv}_k)$ among them or aborts the protocol if this public value does not correspond to one it had in mind.

When initialized with security parameter k, the simulator first generates the CRS for the commitment (public parameters but also extraction and equivocation trapdoors), as well as the possibly required trapdoors to be able to generate, for any pub, a word inside or outside the language $L(\mathsf{pub}, \mathsf{priv})$ when priv is known. It then initializes the real-world adversary $\mathcal{A}$, giving it these values. The simulator then starts its interaction with the environment $\mathcal{Z}$, the functionality $s\mathcal{F}_{\text{LAKE}}$ and its subroutine $\mathcal{A}$.

Since we are in the static-corruption model, the adversary can only corrupt players before the execution of the protocol. We assume players to be honest or not at the beginning, and they cannot be corrupted afterwards. However, this does not prevent the adversary from modifying flows coming from the players. Indeed, since we are in a weak authenticated setting, when a player acts dishonestly (even without being aware of it), it is either corrupted, hence the adversary knows its private values and acts on its behalf; or the adversary tries to impersonate it with chosen/guessed inputs. In both cases, we say the player is $\mathcal{A}$-controlled. Following [CHK+05b], we say that a flow is *oracle-generated* if it was sent by an honest player and arrives without any alteration to the player it was meant to. We say it is *non-oracle-generated* otherwise, that is if it was sent by a $\mathcal{A}$-controlled player (which means corrupted, or which flows have been modified by the adversary). The one-time signatures are aimed at avoiding changes of players during a session: if *pre-flow* is oracle-generated for P_i, then *flow-one* and *flow-three* cannot be non-oracle-generated without causing the protocol to fail because of the signature, for which the adversary does not know the signing key. Similarly, for P_j. On the other hand, if *pre-flow* is non-oracle-generated for P_i, then *flow-one* and *flow-three* cannot be oracle-generated without causing the protocol to fail, since the honest player would sign wrong flows (the flows the player sent before the adversary alters them). In both cases, the verifications of the signatures will fail at Steps $(I3)$ or $(R4)$ and P_i or P_j will abort. One can note that if there is one flow only in the protocol for one player, its signature is not required, which is the case for P_j when there is no pub to agree on at the beginning. But this is just an optimization that can be occasionally applied, as for the PAKE protocol. We do not consider it here.

To deal with both cases of $\mathcal{A}$-controlled players (either corrupted or impersonated by the adversary), we use the Split Functionality model (see Section H.2). We thus add a *pre-flow* which will help us know which players are honest and which ones are $\mathcal{A}$-controlled. If one player is honest and the other one corrupted, the adversary will send the *pre-flow* on behalf of the latter, and the simulator will have to send the *pre-flow* on behalf of the former. But in the case where both players are honest at the beginning of the protocol, both *pre-flow* will have to be sent by $\mathcal{S}$ on behalf of these players and the adversary can then decide to modify one of these flows. This models the fact that the adversary can decide to split a session between P_i and P_j by answering itself to P_i, and thus trying to impersonate P_j with respect to P_i, and doing the same with P_j. Then, the Split Functionality model ensures that two independent sessions are created (with sub-session identifiers). We can thus study these sessions independently, which means that we can assume, right after the *pre-flow*, that either a player is honest if its *pre-flow* is oracle-generated, or $\mathcal{A}$-controlled if the *pre-flow* is non-oracle-generated. Since we want to show that the best possible attack for the adversary (by controlling a player) consists in playing honestly with a trial credential, we have to show that the view of the environment is unchanged if we simulate this dishonest player as an honest player with respect to ideal functionality. The simulator then has to transform its flows into queries to the Ideal Functionality $s\mathcal{F}_{\text{LAKE}}$, and namely the $\mathsf{NewSession}$-query. Still, the $\mathcal{A}$-controlled player is not honest, and can have a bad behavior when sending the real-life flows, but then either it has no strong impact, and it is similar to an honest behavior, or it will make the protocol fail: we cannot avoid the adversary to make denial of service attack, and the adversary will learn nothing.

As explained in [BCL+05] and [ACGP11], where the simulator actually had access to a

TestPwd query to the functionality, it is equivalent to grant the adversary the right to test a password (here a credential) for P_i while trying to play on behalf of P_j (i.e., use a TestPwd query) or to use the split functionality model and generate the NewSession queries corresponding to the $\mathcal{A}$-controlled players and see how the protocol terminates, since it corresponds to a trial of one credential by the adversary (one-line dictionary attack).

The proof will thus consist in generating ideal queries (and namely the NewSession) when receiving non-oracle-generated flows from $\mathcal{A}$-controlled players, and generating real messages for the honest players (whose NewSession queries will be received from the environment). This will be done in a indistinguishable way for the environment.

We assume from now on that we know in which case we are (i.e.how many players are $\mathcal{A}$-controlled), and the pub part is fixed. We then describe the simulator for each of these cases, while it has generated the *pre-flow* for the honest players by generating $(\mathrm{VK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}()$, and thus knows the signing keys. We denote by $L_i = L(\mathsf{pub}, \mathsf{priv}_i)$ the language used by P_i, and by $L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$ the language that P_i expects P_j to use. We use the same notations in the reverse direction. As explained in Section H.1, recall that the languages considered depend on two possibly different relations: $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $L_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. Note that the simulator will use the NewKey query to learn whether the protocol is a success or a failure (in case a player is $\mathcal{A}$-controlled). This will enable it to check whether the LAKE should fulfill, that is, whether the two users play with compatible words and languages, i.e.. $\mathsf{priv}'_i = \mathsf{priv}_i$, $\mathsf{priv}'_j = \mathsf{priv}_j$, $W_i \in L_i$ and $W_j \in L_j$. For the most part, the interaction is implemented by the simulator $\mathcal{S}$ just following the protocol on behalf of all the honest players.

H.D.3 Description of the Simulators

Initialization and Simulation of *pre-flow*.

This is the beginning of the simulation of the protocol, where $\mathcal{S}$ has to send the message *pre-flow* on behalf of each non-corrupted player[2].

Step ($I0$). When receiving the first $(\mathsf{Init} : \mathrm{ssid}, P_i, P_j, \mathsf{pub}_i)$ from $s\mathcal{F}_{\mathrm{LAKE}}$ as answer to the Init query sent by the environment on behalf of P_i, $\mathcal{S}$ starts simulating the new session of the protocol for party P_i, peer P_j, session identifier ssid. $\mathcal{S}$ chooses a key pair $(\mathrm{SK}_i, \mathrm{VK}_i)$ for a one-time signature scheme and generates a *pre-flow* message with the values $(\mathrm{VK}_i, \mathsf{pub}_i)$. It gives this message to $\mathcal{A}$ on behalf of (P_i, ssid).

Step ($R0$). When receiving the second $(\mathsf{Init} : \mathrm{ssid}, P_j, P_i, \mathsf{pub}_j)$ from $s\mathcal{F}_{\mathrm{LAKE}}$ as answer to the Init query sent by the environment on behalf of P_j, $\mathcal{S}$ starts simulating the new session of the protocol for party P_j, peer P_i, session identifier ssid. $\mathcal{S}$ chooses a key pair $(\mathrm{SK}_j, \mathrm{VK}_j)$ for a one-time signature scheme and generates a *pre-flow* message with the values $(\mathrm{VK}_j, \mathsf{pub}_j)$. It gives this message to $\mathcal{A}$ on behalf of (P_j, ssid).

Splitting the Players.

As just said, thanks to the Split Functionality model, according to which flows were transmitted or altered by $\mathcal{A}$, we know from the *pre-flow* which player(s) is (are) honest and which player(s) is (are) $\mathcal{A}$-controlled, and the public part pub. We can consider each case independently after the initial split, during which $\mathcal{S}$ generated the signing keys of the honest players. Thanks to the signature in the last flows for each player, if the adversary tries to take control on behalf of a honest user for some part of the execution (without learning the internal states, since we

[2]Note that $\mathcal{S}$ only has to send one of these flows if one player is corrupted.

exclude adaptive corruptions), the verification will fail. Then we can assume that the sent flows are the received flows.

One can note that the prior agreement on pub allows to simulate P_i before having received any information from P_j, and also without knowing whether the protocol should be a success or not. Without such an agreement, the simulator would not know which value to use for pub whereas it cannot change its mind later, since it is sent in clear. Everything else is committed: either in an equivocable way on behalf of P_i so that we can change it later when we know the real status of the session; or in a non-equivocable way on behalf of P_j since we can check the status of the session before making this commitment. Of course, both commitments are extractable. In the whole proof, in case the extraction fails, the simulator acts as if the simulation should fail. Indeed, the language of the smooth projective hash function not only verifies the equations, but also that the ciphertext is valid, and this verification will fail.

We come back again to the case of our equivocable commitment with SPHF that is not a really extractable/binding commitment since the player can open it in a different way one would extract it, in case the second ciphertext does not encrypt $1_{\mathbb{G}}$: if extraction leads to an inconsistent tuple, there is little chance that with the random $\vec{\varepsilon}$ it becomes consistent; if extraction leads to a consistent tuple, there is little chance that with the random $\vec{\varepsilon}$ it remains consistent, and then the real-life protocol will fail, whereas the ideal-one was successful at the NewKey-time. But then, because of the positive NewKey-answer, the $\mathsf{SendKey}$-query takes the key-input into consideration, that is random on the initiator side because of the SPHF on an invalid word, and thus indistinguishable from the environment point of view from a failed session: this is a denial of service, the adversary should already be aware of.

Hence, the three simulations presented below exploit the properties of our commitments and SPHF to make the view of the environment indistinguishable from a real-life attack, just using the simulator $\mathcal{S}$ that is allowed to interact with the ideal functionality on behalf of players, but in an honest way only, since the functionality is perfect and does not know bad behavior.

During all these simulations, $\mathcal{S}$ knows the equivocability trapdoor of the commitment and the decryption keys of the two encryption schemes.

Case 1: P_i is $\mathcal{A}$-controlled and P_j is honest.

In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_j, for which it has simulated the *pre-flow* and thus knows the signing key, and has to simulate the queries to the functionality as if the $\mathcal{A}$-controlled player P_i was honest.

Step $(I1)$. This step is taken care of by the adversary, who sends its *flow-one*, from which $\mathcal{S}$ extracts $(\mathsf{priv}_i, \mathsf{priv}'_j)$ only. No need to extract W_i, but one generates a random valid $V_i \in L(\mathsf{pub}, \mathsf{priv}_i)$ (we have assumed the existence of a trapdoor in the CRS to generate such valid words). $\mathcal{S}$ then sends the query

$$(\mathsf{NewSession} : \mathrm{ssid}', P_i, P_j, V_i, L_i = L(\mathsf{pub}, \mathsf{priv}_i), L'_j = L(\mathsf{pub}, \mathsf{priv}'_j), \mathsf{initiator})$$

to $\mathcal{F}_{\text{LAKE}}$ on behalf of P_i.

Step $(R2)$. The $\mathsf{NewSession}$ query for this player (P_j, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\text{LAKE}}$ to $\mathcal{F}_{\text{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_j, P_i, \mathsf{pub}, \mathsf{receiver})$ and makes a call NewKey to the functionality to check the success of the protocol. It actually tells whether the languages are consistent, but does not tell anything about the validity of the word submitted by the adversary for P_i. It indeed receives the answer in the name of P_i. In case of a success, $\mathcal{S}$ generates a word $V_j \in L(\mathsf{pub}, \mathsf{priv}'_j)$ and uses $\mathsf{priv}_j = \mathsf{priv}'_j$ and $\mathsf{priv}'_i = \mathsf{priv}_i$ for this receiver session (we have assumed the existence of a trapdoor in the CRS to generate such valid words) and produces

a commitment $\mathcal{C}_j$ on the tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. Otherwise, $\mathcal{S}$ produces a commitment $\mathcal{C}_j$ on a dummy tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. It then generates a challenge value $\vec{\varepsilon}$ and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the *flow-two* message $(\mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ to $\mathcal{A}$ on behalf of P_j, where σ_j is the signature on all the previous information.

Step $(I3)$. This step is taken care of by the adversary, who sends its *flow-three*.

Step $(R4)$. Upon receiving $m = (\textit{flow-three}, \mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$, $\mathcal{S}$ makes the verification checks, and possibly aborts. In case of correct checks, $\mathcal{S}$ already knows whether the protocol should succeed, thanks to the NewKey query. If the protocol is a success, then $\mathcal{S}$ computes receiver session key honestly, and makes a SendKey to P_j. Otherwise, $\mathcal{S}$ makes a SendKey to P_j with a random key that will anyway not be used.

Case 2: P_i is honest and P_j is $\mathcal{A}$-controlled.

In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_i, for which it has simulated the *pre-flow* and thus knows the signing key, and has to simulate the queries to the functionality as if the $\mathcal{A}$-controlled player P_j was honest.

Step $(I1)$. The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\text{LAKE}}$ to $\mathcal{F}_{\text{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ and generates a *flow-one* message by committing to a dummy tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$. It gives this commitment $(\mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid').

Step $(R2)$. This step is taken care of by the adversary, who sends its *flow-two* $= (\textit{flow-two}, \mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$, from which $\mathcal{S}$ first checks the signature, and thereafter extracts the committed triple $(\mathsf{priv}_j, \mathsf{priv}'_i, W_j)$. $\mathcal{S}$ then sends the query

$$(\mathsf{NewSession} : \mathrm{ssid}', P_j, P_i, W_j, L_j = L(\mathsf{pub}, \mathsf{priv}_j), L'_i = L(\mathsf{pub}, \mathsf{priv}'_i), \mathsf{receiver})$$

to $\mathcal{F}_{\text{LAKE}}$ on behalf of P_j.

Step $(I3)$. $\mathcal{S}$ makes a NewKey query to the functionality to know whether the protocol should succeed. It indeed receives the answer in the name of P_j. In case of a success, $\mathcal{S}$ generates a word $V_i \in L(\mathsf{pub}, \mathsf{priv}'_i)$ and uses $\mathsf{priv}_i = \mathsf{priv}'_i$ for this initiator session (we have assumed the existence of a trapdoor in the CRS to generate such valid words) and then uses the equivocability trapdoor to update $\mathcal{C}'_i$ and t in order to contain the new consistent tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ with respect to the challenge $\vec{\varepsilon}$. If the protocol should be a success, then $\mathcal{S}$ computes initiator session key honestly, and makes a SendKey to P_i. Otherwise, $\mathcal{S}$ makes a SendKey to P_i with a random key that will anyway not be used. $\mathcal{S}$ sends the *flow-three* message $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

Step $(R4)$. This step is taken care of by the adversary.

Case 3: P_i and P_j are honest.

In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the two honest players P_i and P_j, for which it has simulated the *pre-flow* and thus knows the signing keys. But since no player is controlled by $\mathcal{A}$, the NewKey query will not provide any answer to the simulator. But thanks to the semantic security of the commitments, dummy values can be committed, no external adversary will make any difference.

Step $(I1)$. The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\text{LAKE}}$ to $\mathcal{F}_{\text{LAKE}}$ (transforming the session identifier from ssid to

ssid′). $\mathcal{S}$ receives the answer (NewSession : ssid, P_i, P_j, pub, initiator) and generates a *flow-one* message by committing to a dummy tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$. It gives this commitment $(\mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid').

Step $(R2)$. The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\text{LAKE}}$ to $\mathcal{F}_{\text{LAKE}}$ (transforming the session identifier from ssid to ssid′). $\mathcal{S}$ receives the answer (NewSession : ssid, P_j, P_i, pub, receiver) and generates a commitment $\mathcal{C}_j$ on a dummy tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. It then generates a challenge value $\vec{\varepsilon}$ and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the *flow-two* message $(\mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ to $\mathcal{A}$ on behalf of P_j, where σ_j is the signature on all the previous information.

Step $(I3)$. When the session $(P_i; \text{ssid}')$ receives the message $m = (\textit{flow-two}, \mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ from its peer session $(P_j; \text{ssid}')$, the signature is necessarily correct. Then, $\mathcal{S}$ makes a SendKey to P_i with a random key that will anyway not be used, since no player is corrupted. $\mathcal{S}$ sends the *flow-three* message $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

Step $(R4)$. When the session $(P_j; \text{ssid}')$ receives the message $m = (\textit{flow-three}, \mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ from its peer session $(P_i; \text{ssid}')$, the signature is necessarily correct. $\mathcal{S}$ makes a SendKey to P_j with a random key that will anyway not be used, since no player is corrupted.

H.D.4 Description of the Games

We now provide the complete proof by a sequence of games, where we replace the triple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ by the notation T_i, and the triple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$ by the notation T_j, with component-wise operations to simplify notations. Similarly, for cleaner notations, we use non-vector notations for the ciphertexts, the random coins and the challenge ε, but all the computations are assumed to be performed component-wise, and thus implicitly use vectors.

We insist that we are considering static corruptions only, and with the split-functionality, we already know which players are corrupted and verification keys for the one-time signatures are known to the two players, and fixed: either honestly generated (honest player) or adversary-generated (corrupted players).

Game $\boldsymbol{G}_0$: This is the real game, where every flow from honest players are generated correctly by the simulator which knows the inputs sent by the environment to the players. There is no use of the ideal functionality for the moment.

Game $\boldsymbol{G}_1$: In this game, the simulator knows the decryption key for $\mathcal{C}_i$ when generating the CRS. But this game is almost the same as the previous one except the way sk_j is generated when P_i is corrupted and P_j honest. In all the other cases, the simulator does as in $\boldsymbol{G}_0$ by playing honestly (still knowing its private values). When P_i is corrupted and P_j honest, $\mathcal{S}$ does as before until $(R4)$, but then, it extracts the values committed to by the adversary in Com_i (using the decryption key for $\mathcal{C}_i$) and checks whether the private parts of the languages are consistent with the values sent to P_j by the environment. If the languages are not consistent (or decryption rejects), P_j is given a random session key sk_j.

This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF on Com_i.

Game $\boldsymbol{G}_2$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. This game is almost the same as the previous one except that $\mathcal{S}$ extracts the values committed to by the adversary in $\mathcal{C}_i$ to check consistency of the languages, and does not wait until Com_i. If the languages are not consistent (or decryption rejects), P_j is given a random session key sk_j.

The game is indistinguishable from the previous one except if Com_i contains consistent values whereas $\mathcal{C}_i$ does not, but because of the unpredictability of ε, and the Pedersen commitment that is computationally binding under the discrete logarithm problem, the probability is bounded by $1/q$.

The distance between the two games is thus bounded by the probability to break the binding property of the Pedersen commitment.

Game $\boldsymbol{G}_3$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Actually, in the above game, when P_i is corrupted and P_j honest, if extracted languages from $\mathcal{C}_i$ are not consistent, P_j does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}_j$ with dummy values T'_j.

This game is computationally indistinguishable from the former one thanks to the IND-CPA property of the encryption scheme involved in $\mathcal{C}_j$. To prove this indistinguishability, one makes q hybrid games, where q is the number of such sessions where P_i is corrupted and P_j is honest but extracted languages from $\mathcal{C}_i$ are not consistent with inputs to P_j. More precisely, in the k-th hybrid game G_k (for $1 \leq k \leq q$), in all such sessions before the k-th one, $\mathcal{C}_j$ is generated by encrypting T'_j, in all sessions after the k-th one, $\mathcal{C}_j$ is generated by encrypting T_j, and in the k-th session, $\mathcal{C}_j$ is generated by calling the left-or-right encryption oracle on (T_j, T'_j). It is clear that the game $\boldsymbol{G}_2$ correponds to G_1 with the "left" oracle, and the game $\boldsymbol{G}_3$ corresponds to G_q with the "right" oracle. And each time, G_k with the right oracle is identical to G_{k+1} with the "left" oracle, while every game G_k is an IND-CPA game. It is possible to use the encryption oracle because the random coins are not needed in these sessions.

Game $\boldsymbol{G}_4$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Now, when P_i is corrupted and P_j honest, if extracted languages from $\mathcal{C}_i$ are consistent, $\mathcal{S}$ knows priv_j and priv'_i (the same as the values sent by the environment). It furthermore generates a random valid word V_j, and uses it to generate the ciphertext $\mathcal{C}_j$ instead of re-randomizing the word W_j sent by the environment. $\mathcal{S}$ can compute the correct value sk_j from the random coins, and gives it to P_j.

This game is perfectly indistinguishable from the former one thanks to the self-randomizable property of the language.

Note that the value sk_j computed by $\mathcal{S}$ can be computed by the adversary if the latter indeed sent a valid word W_i in $\mathcal{C}_i$ (that is not explicitly checked in this game). Otherwise, sk_j looks random from the smoothness of the SPHF. As a consequence, on this game, sessions where P_i is corrupted and P_j is honest look ideal, while one does not need anymore the inputs from the environment sent to P_j to simulate honest players.

Game $\boldsymbol{G}_5$: We now consider the case where P_i is honest. The simulator has to simulate P_i behavior. To do so, it will know the equivocability trapdoor for the Pedersen commitment. But for other cases, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. In $(I1)$, the simulator still encrypts $T_i = (\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ from the environment to produce $\mathcal{C}_i$. It chooses at random a dummy value $\mathcal{C}'_i$ and computes honestly the equivocable commitment $\mathcal{C}''_i$, knowing the random value t_i. In $(I3)$, after receiving ε from P_j, it chooses random coins z_i and computes Com_i as the encryption of $T_i = (\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ with the random coins z_i. (Since this is a double encryption scheme, it uses the redundancy from $\mathcal{C}_i$: namely for DLCS, it uses ξ from $\mathcal{C}_i$). Thanks to the homomorphic property, it can compute $\mathcal{C}'_i$ as $(\mathsf{Com}_i/\mathcal{C}_i)^{1/\varepsilon}$, and equivocate $\mathcal{C}$"$_i$. $\mathcal{C}'_i$ should be an encryption of $1_{\mathbb{G}}$ under the random coins r'_i that are implicitly defined, but unknown.

Thanks to the properties of the different commitments recalled in Section H.D.1, and the perfect-hiding property of the Pedersen commitment, this is a perfect simulation. It then com-

putes the hash values honestly, using z_i.

Game $\boldsymbol{G}_6$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing dummy values T'_i instead of T_i. Everything else is unchanged from $\boldsymbol{G}_5$.

This game is thus indistinguishable from the former one thanks to the IND-CCA property of the encryption scheme involved in $\mathcal{C}_i$. As for the proof of indistinguishability of Game $\boldsymbol{G}_3$, we do a sequence of hybrid games, where $\mathcal{C}_i$ is generated be either encrypting T_i or T'_i, or asking the left-or-right oracle on (T_i, T'_i). We replace the decryption key for $\mathcal{C}_i$ by access to the decryption oracle on $\mathcal{C}_i$. Then, one has to take care that no decryption query is asked on one of the challenge ciphertexts involved in the sequence of games. This would mean that the adversary would replay in another session a ciphertext oracle-generated in another session. Because of the label which contains the verification key oracle-generated, one can safely reject the ciphertext.

Game $\boldsymbol{G}_7$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing dummy values T'_i. It then computes $\mathcal{C}'_i$ by encrypting the value $(T_i/T'_i)^{1/\varepsilon}$ with randomness $z_i - r_i/\varepsilon$. This leads to the same computations of $\mathcal{C}_i$ and $\mathcal{C}'_i$ as in the former game. The rest is done as above.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_8$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i and P_j are both honest (both initiation flows where oracle-generated), if the words and languages are correct, players are both given the same random session key $\mathsf{sk}_i = \mathsf{sk}_j$. If the words and languages are not compatible, random independent session keys are given.

Since the initiation flows ($I0$ and $R0$) contained oracle-generated verification keys, unless the adversary managed to forge signatures, all the flows are oracle-generated. First, because of the pseudo-randomness of the SPHF, H_i is unpredictable, and independent of H'_j, hence sk_i looks random. Then, if the words and languages are compatible, we already has $\mathsf{sk}_j = \mathsf{sk}_i$ in the previous game. However, if they are not compatible, either H'_i is independent of H_i, or H'_j is independent of H_j, and in any case, sk_j where already independent of sk_i in the previous game.

This game is thus computationally indistinguishable from the former one, under the pseudo-randomness of the two SPHF.

Game $\boldsymbol{G}_9$: In this above game, the hash values do not have to be computed anymore when P_i and P_j are both honest. The random coins are not needed anymore.

In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i and P_j are both honest, $\mathcal{S}$ generates $\mathcal{C}'_i$ and $\mathcal{C}_j$ with dummy values T'_i and T'_j. In this game, sessions where P_i and P_j are both honest look ideal, while one does not need anymore the inputs from the environment sent to P_i and P_j to simulate honest players.

This game is computationally indistinguishable from the former one thanks to the IND-PD-CCA and IND-CPA properties of the encryption schemes involved in $\mathcal{C}'_i$ and $\mathcal{C}_j$. For the proof on indistinguishability between the two games, we make two successive sequences of hybrid games, as for the proof of indistinguishability of Game $\boldsymbol{G}_3$. One with the IND-PD-CCA game: a sequence of hybrid games, where $\mathcal{C}_i$ is generated by encrypting T'_i, and $\mathcal{C}'_i$ by encrypting either T_i or T'_i, but in the critical session, one asks for the left-or-right oracle Encrypt on (T'_i, T'_i), and the left-or-right oracle Encrypt′ on (T_i, T'_i). The decryption key for $\mathcal{C}_i$ is replaced by an access to the decryption oracle on $\mathcal{C}_i$. As above, one has to take care that no decryption query is asked on a challenge ciphertext $\mathcal{C}'_i$, but the latter cannot be valid since it is computed from

$\mathcal{C}_i$ values not controlled by the adversary. The second hybrid sequence uses IND-CPA games on $\mathcal{C}_j$ exactly as in the proof of indistinguishability of Game $\boldsymbol{G}_3$.

Game $\boldsymbol{G}_{10}$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS, but also the decryption key for $\mathcal{C}_j$. When P_i is honest and P_j corrupted, $\mathcal{S}$ extracts the values committed to by the adversary in $\mathcal{C}_j$. It checks whether they are consistent with the values sent to P_i by the environment. If the words and languages are not consistent (or decryption rejects), P_i is given a random session key sk_i.

This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G}_{11}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS.

In the above game, when P_i is honest and P_j corrupted, if extracted values from $\mathcal{C}_j$ are not consistent, P_i does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}'_i$ with dummy values T'_i.

This game is computationally indistinguishable from the former one thanks to the IND-PD-CCA property of the encryption scheme involved in $\mathcal{C}'_i$. The proof uses the same sequence of hybrid games with the IND-PD-CCA game on $(\mathcal{C}_i, \mathcal{C}'_i)$ as in the proof of indistinguishability of Game $\boldsymbol{G}_9$.

Game $\boldsymbol{G}_{12}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. Now, when P_i is honest and P_j corrupted, if extracted values from $\mathcal{C}_j$ are consistent, $\mathcal{S}$ knows priv_i and priv'_j (the same as the values sent by the environment). It furthermore generates a random valid word V_i, and uses it to generate the ciphertext $\mathcal{C}'_i$ instead of re-randomizing the word W_j sent by the environment. $\mathcal{S}$ can compute the correct value sk_i from the random coins, and gives it to P_i. In this game, sessions where P_i is honest and P_j is corrupted look ideal, while one does not need anymore the inputs from the environment sent to P_i to simulate honest players.

This game is perfectly indistinguishable from the former one thanks to the self-randomizable property of the language.

Game $\boldsymbol{G}_{13}$: In this game, $\mathcal{S}$ now uses the ideal functionality: NewSession-queries for honest players are automatically forwarded to the ideal functionality, for corrupted players, they are done by $\mathcal{S}$ using the values extracted from $\mathcal{C}_i$ or $\mathcal{C}_j$. In order to check consistency of the words and languages, $\mathcal{S}$ asks for a NewKey. When one player is corrupted, it learns the outcome: success or failure. It can continue the simulation in an appropriate way.

H.E Complexity

In the Table H.1, we give the number of elements to be sent (group elements or scalars) and the number of exponentiations to compute for each operation (commitment and SPHF), where we consider the Equality Test, and the Linear Pairing Product Equations. One has to commit all the private inputs, and then the cost for relations is just the additional overhead due to the projection keys and hashing computations once the elements are already committed: an LCSCom commitment is 5 group elements, and a DLCSCom$'$ is twice more, plus the Pedersen commitment (one group element), the challenge ε (a scalar) and the opening t (a scalar). Note that a simple Linear commitment is just 3 group elements.

If the global language is a conjunction of several languages, one should simply add all the costs, and consider the product of all the sub-hashes as the final hash value from the SPHF.

DLin	$\mathbb{G}$	$\mathbb{Z}_p$	Exp.	CSCom	$\mathbb{G}$	$\mathbb{Z}_p$	Exp.
LCSCom	$5n$	0	$7n+2$	CSCom	$4n$	0	$4n+1$
DLCSCom	$10n+1$	2	$18n+6$	DCSCom	$8n+1$	2	$12n+5$
Equality	2	0	14	Equality	1	0	10
LPPE	$2n+1$	0	$10n+11$	LPPE	$n+1$	0	$7n+9$

Table H.1: Computational and Communication Costs

PAKE.

Two users want to prove to each other they possess the same password. In this case $W_i = \mathsf{priv}'_j = \mathsf{priv}_i = \mathsf{priv}_j = \mathsf{priv}'_i = W_j$. So P_i will commit to his password, and thus a unique DLCSCom commitment for W_i, priv_i and priv'_i. P_j can use a simple Linear commitment. They then send projection keys for equality tests: 13 group elements and 2 scalars for Com_i and 5 group elements for Com_j, plus VK_i and σ_i. This leads to 18 group elements and two scalars our PAKE scheme. The DDH-based variant would use 11 group elements and 2 scalars only in total, which is far more efficient than existing solutions, and namely [ACP09] that uses a bit-per-bit commitment to provide equivocability.

Verifier-based PAKE.

As explained earlier, we do a PAKE with the common password $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, where h has been chosen by the server: the commitment Com_i needs 21 group elements plus 2 scalars, and 4 additional group elements to check it; the commitment Com_j needs 6 group elements, and 4 additional elements to check it. Because of the ephemeral h, one has to send in total 35 group elements and 2 scalars, plus the one-time signatures. The DDH-based variant would use 25 group elements and 2 scalars only in total.

Secret Handshake.

The users want to check their partner possesses a valid signature on their public identity or pseudonym (in pub) under some valid but private verification key (affiliation-hiding). More precisely, P_i wants to prove he possesses a valid signature σ on the public message m (his identity or a pseudonym) under a private verification key vk: we thus have m in the pub part, $\mathsf{priv}_i = \mathsf{vk}$ and $W = \sigma$. This is the same for P_j. Using Waters signature, $\sigma = (\sigma_1, \sigma_2)$, where σ_1 only has to be encrypted, because σ_2 does not contain any information, it can thus be sent in clear. In addition, as noticed from the security proof, σ_2 does not need to be encrypted in an IND-PD-CCA manner, but with a simple IND-CPA encryption scheme in the third round. To achieve unlinkability, one can rerandomize this signature σ to make the σ_2 values different and independent each time.

As a consequence, the committed values are: vk that can be any group element, since with the master secret key s such that $h = g^s$ for the global parameters of the Waters signature (see the Appendix H.A.3) one can derive the signing key associated to any verification key, and thus generate a valid word in the language; and σ_1 in IND-CPA only. One additionally sends σ_2 in clear, and so 14 group elements plus 2 scalars for Com_i, and 7 group elements for Com_j. The languages to be verified are $\mathsf{priv}_i = \mathsf{priv}'_i$, on the committed $\mathsf{priv}_i = \mathsf{vk}_i$ with the expected $\mathsf{priv}'_i = \mathsf{vk}'_i$, and the Linear Pairing Product Equation for the committed signature σ_i, but under the expected vk'_i: 5 group elements for the projection keys in both directions: 31 group elements plus 2 scalars are sent in total.

Execution between P_i and P_j, with session identifier sid.

- Preliminary Round: each user generates a pair of signing/verification keys $(\mathsf{SK}, \mathsf{VK})$ and sends VK together with its contribution to the public part of the language.

We denote by $\ell_i = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_i, \mathsf{VK}_j)$ and by $\ell_j = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_j, \mathsf{VK}_i)$, where pub is the combination of the contributions of the two players. The initiator now uses a word W_i in the language $L(\mathsf{pub}, \mathsf{priv}_i)$, and the receiver uses a word W_j in the language $L(\mathsf{pub}, \mathsf{priv}_j)$, possibly re-randomized from their long-term secrets*. We assume commitments and associated smooth projective hash functions exist for these languages.

- First Round: user P_i (with random tape ω_i) generates a multi-$\mathsf{DLCSCom}'$ commitment on $(\mathsf{priv}_i, \mathsf{priv}'_j, W_i)$ in $(\mathcal{C}_i, \mathcal{C}'_i)$, where W_i has been randomized in the language, under the label ℓ_i. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: user P_j (with random tape ω_j) computes a multi-LCS commitment on $(\mathsf{priv}_j, \mathsf{priv}'_i, W_j)$ in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\vec{r}$, where W_j has been randomized in the language, under the label ℓ_j. It then generates a challenge $\vec{\varepsilon}$ on $\mathcal{C}_i$ and hashing/projection keys† hk_i and hp_i associated to $\mathcal{C}_i$ (which will be associated to the future Com_i). It finally signs all the flows using SK_j in σ_j, and sends $(\mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ to P_i;
- Third Round: user P_i first checks the signature σ_j, computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}_i'^{\vec{\varepsilon}}$ and witness $\mathbf{z}$ (from $\vec{\varepsilon}$ and ω_i), it generates hashing/projection keys hk_j and hp_j associated to Com_j. It finally signs all the flows using SK_i in σ_i, and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature σ_i and the correct opening of $\mathcal{C}''_i$ into $\mathcal{C}'_i$, it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}_i'^{\vec{\varepsilon}}$.

  P_i computes K_i and P_j computes K_j as follows:

$$\begin{aligned} K_i &= \mathsf{Hash}(\mathsf{hk}_j, \{(\mathsf{priv}'_j, \mathsf{priv}_i)\} \times L(\mathsf{pub}, \mathsf{priv}'_j), \ell_j, \mathsf{Com}_j) \\ &\quad \cdot \mathsf{ProjHash}(\mathsf{hp}_i, \{(\mathsf{priv}_i, \mathsf{priv}'_j)\} \times L(\mathsf{pub}, \mathsf{priv}_i), \ell_i, \mathsf{Com}_i; \mathbf{z}) \\ K_j &= \mathsf{ProjHash}(\mathsf{hp}_j, \{(\mathsf{priv}_j, \mathsf{priv}'_i)\} \times L(\mathsf{pub}, \mathsf{priv}_j), \ell_j, \mathsf{Com}_j; \vec{r}) \\ &\quad \cdot \mathsf{Hash}(\mathsf{hk}_i, \{(\mathsf{priv}'_i, \mathsf{priv}_j)\} \times L(\mathsf{pub}, \mathsf{priv}'_i), \ell_i, \mathsf{Com}_i) \end{aligned}$$

*As explained in Section H.1, recall that the languages considered depend on two possibly different relations, namely $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $L_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. We assume they are both self-randomizable.

†Recall that the SPHF is constructed in such a way that this projection key does not depend on $\mathcal{C}'_i$ and is indeed associated to the future whole Com_i.

Figure H.4: Language-based Authenticated Key Exchange from a Smooth Projective Hash Function on Commitments

P_i uses a password W_i and P_j uses a password W_j. We denote $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

- First Round: P_i (with random tape ω_i) first generates a pair of signing/verification keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ and a $\mathsf{DLCSCom}'$ commitment on W_i in $(\mathcal{C}_i, \mathcal{C}'_i)$, under $\ell_i = (\ell, \mathsf{VK}_i)$. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathsf{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: P_j (with random tape ω_j) computes a LCSCom commitment on W_j in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\vec{r}$, under the label ℓ. It then generates a challenge ε on $\mathcal{C}_i$ and hashing/projection keys hk_i and the corresponding hp_i for the equality test on Com_i ("Com_i is a valid commitment of W_j", this only requires the value ξ_i computable thanks to $\mathcal{C}_i$). It then sends $(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i)$ to P_i;
- Third Round: user P_i can compute $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$ and witness $\mathbf{z}$ (from ε and ω_i), it generates hashing/projection keys hk_j and hp_j for the equality test on Com_j. It finally signs all the flows using SK_i in σ_i and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature and the validity of the Pedersen commitment (thanks to t), it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$. P_i computes K_i and P_j computes K_j as follows:

$$\begin{aligned} K_i &= \mathsf{Hash}(\mathsf{hk}_j, L'_j, \ell, \mathsf{Com}_j) \cdot \mathsf{ProjHash}(\mathsf{hp}_i, L_i, \ell_i, \mathsf{Com}_i; \mathbf{z}) \\ K_j &= \mathsf{ProjHash}(\mathsf{hp}_j, L_j, \ell, \mathsf{Com}_j; \vec{r}) \cdot \mathsf{Hash}(\mathsf{hk}_i, L'_i, \ell_i, \mathsf{Com}_i) \end{aligned}$$

Figure H.5: Password-based Authenticated Key Exchange

- $\mathsf{Setup}(1^k)$: A group $\mathbb{G}$ of prime order p, with ten independent generators $(g_1, g_2, g_3, h_1, h_2, c_1, c_2, d_1, d_2, \zeta) \xleftarrow{\$} \mathbb{G}^{10}$, a collision-resistant hash function $\mathfrak{H}_K$, and possibly an additional reversible mapping $\mathcal{G}$ from $\{0,1\}^k$ to $\mathbb{G}$ to commit to bit-strings. One can denote $\mathsf{ek} = (c_1, c_2, d_1, d_1, h_1, h_2, \mathfrak{H}_K)$;
- $\mathsf{Commit}(\ell, \vec{M}; \vec{r}, \vec{s}, \vec{a}, \vec{b}, t)$: for $(\vec{r}, \vec{s}, \vec{a}, \vec{b}, t) \xleftarrow{\$} \mathbb{Z}_p^{4n+1}$

$$\begin{array}{lll} (\mathcal{C}, \mathcal{C}') \leftarrow n - \mathsf{DLCS}(\ell, \mathsf{ek}, \vec{M}, (1_{\mathbb{G}})^n; \vec{r}, \vec{s}, \vec{a}, \vec{b}) & & \\ \chi = \mathfrak{H}_K(\vec{M}, \mathcal{C}'), \mathcal{C}'' = g_1^t \zeta^\chi & \xrightarrow{\quad \mathcal{C}, \mathcal{C}'' \quad} & \\ \prod_i \varepsilon_i \stackrel{?}{\neq} 0 \bmod p & \xleftarrow{\quad \vec{\varepsilon} \quad} & \varepsilon \xleftarrow{\$} \mathbb{Z}_p^*, \\ \vec{z} = (\vec{r} + \vec{\varepsilon} \cdot \vec{a} \bmod p, \vec{s} + \vec{\varepsilon} \cdot \vec{b} \bmod p) & & \vec{\varepsilon} \leftarrow (\varepsilon, \ldots, \varepsilon) \\ \textsc{Erase}(\vec{r}, \vec{s}, \vec{a}, \vec{b}) & & \end{array}$$

- $\mathsf{Decommit}(\ell, \mathcal{C}, \mathcal{C}', \vec{\varepsilon})$:

$$\begin{array}{lll} & \xrightarrow{\quad \mathcal{C}', t, \vec{M}, \mathbf{z} \quad} & \text{compute } \xi \text{ from } \mathcal{C} \\ & & \chi = \mathfrak{H}_K(\vec{M}, \mathcal{C}'), \mathcal{C}'' \stackrel{?}{=} g_1^t \zeta^\chi \\ & & \mathcal{C} \cdot \mathcal{C}'^{\vec{\varepsilon}} \stackrel{?}{=} n - \mathsf{LCS}^*(\ell, \vec{M}, \xi; \mathbf{z}_r, \mathbf{z}_s) \end{array}$$

Figure H.6: $n - \mathsf{DLCS}$ Commitment Scheme

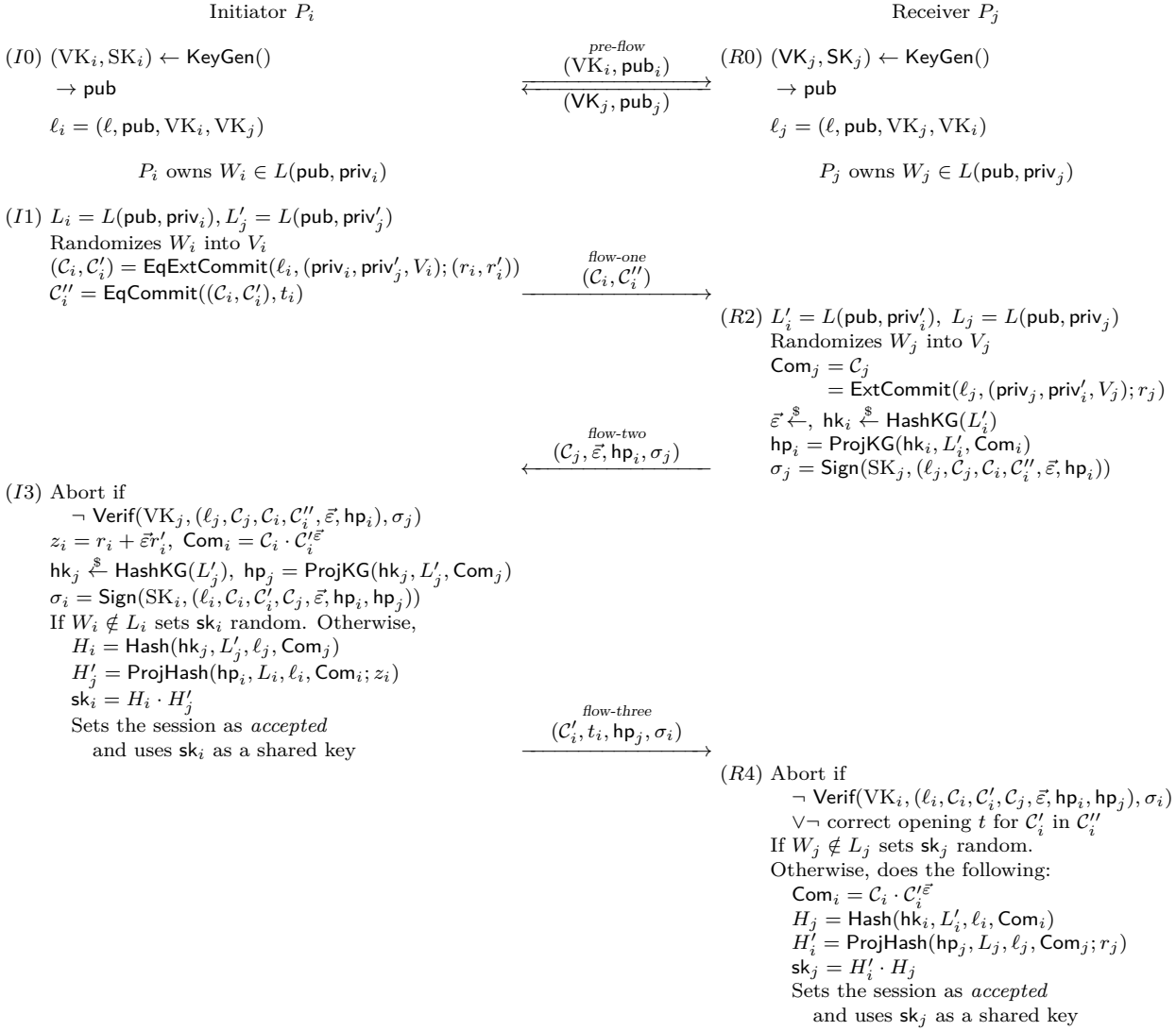


Figure H.7: Description of the language authenticated key exchange protocol for players (P_i, ssid), with index i, message $W_i \in L_i = L(\mathsf{pub}, \mathsf{priv}_i)$ and expected language for P_j $L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$ and (P_j, ssid), with index j, message $W_j \in L_j = L(\mathsf{pub}, \mathsf{priv}_j)$ and expected language for P_i $L'_i = L(\mathsf{pub}, \mathsf{priv}'_i)$. The label is $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$. The random values used in the commitments (witnesses) are all included in (r_i, r'_i) and r_j.

Appendix I

New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange

Crypto 2013
[BBC+13b] with F. Benhamouda, O. Blazy, C. Chevalier and D. Pointcheval

Abstract : *Password-Authenticated Key Exchange* (PAKE) has received deep attention in the last few years, with a recent improvement by Katz and Vaikuntanathan, and their one-round protocols: the two players just have to send simultaneous flows to each other, that depend on their own passwords only, to agree on a shared high entropy secret key. To this aim, they followed the Gennaro and Lindell's approach, with a new kind of *Smooth-Projective Hash Functions* (SPHFs). They came up with the first concrete one-round PAKE, secure in the Bellare, Pointcheval, and Rogaway's model, but at the cost of a simulation-sound NIZK, which makes the overall construction not really efficient.

This paper follows their path with a new efficient instantiation of SPHF on Cramer-Shoup ciphertexts. It then leads to the design of the most efficient PAKE known so far: a one-round PAKE with two simultaneous flows consisting of 6 group elements each only, in any DDH-group without any pairing. We thereafter show a generic construction for SPHFs, in order to check the validity of complex relations on encrypted values. This allows to extend this work on PAKE to the more general family of protocols, termed *Langage-Authenticated Key Exchange* (LAKE) by Ben Hamouda, Blazy, Chevalier, Pointcheval, and Vergnaud, but also to blind signatures. We indeed provide the most efficient blind Waters' signature known so far.

I.1 Introduction

Authenticated Key Exchange

protocols are quite important primitives for practical applications, since they enable two parties to generate a shared high entropy secret key, to be later used with symmetric primitives in order to protect communications, while interacting over an insecure network under the control of an adversary. Various authentication means have been proposed, and the most practical one is

definitely a shared low entropy secret, or a password they can agree on over the phone, hence PAKE, for *Password-Authenticated Key Exchange*. The most famous instantiation has been proposed by Bellovin and Merritt [BM92], EKE for Encrypted Key Exchange, which simply consists of a Diffie-Hellman key exchange [DH76], where the flows are symmetrically encrypted under the shared password. Overall, the equivalent of 2 group elements have to be sent.

A first formal security model was proposed by Bellare, Pointcheval and Rogaway [BPR00] (the BPR model), to deal with off-line dictionary attacks. It essentially says that the best attack should be the on-line exhaustive search, consisting in trying all the passwords by successive executions of the protocol with the server. Several variants of EKE with BPR-security proofs have been proposed in the ideal-cipher model or the random-oracle model [Poi12]. Katz, Ostrovsky and Yung [KOY01] proposed the first practical scheme (KOY), provably secure in the standard model under the DDH assumption. This is a 3-flow protocol, with the client sending 5 group elements plus a verification key and a signature, for a one-time signature scheme, and the server sending 5 group elements. It has been generalized by Gennaro and Lindell [GL03] (GL), making use of smooth projective hash functions.

Smooth Projective Hash Functions

(SPHFs) were introduced by Cramer and Shoup [CS02] in order to achieve IND-CCA security from IND-CPA encryption schemes, which led to the first efficient IND-CCA encryption scheme provably secure in the standard model under the DDH assumption [CS98]. They can be seen as a kind of implicit designated-verifier proofs of membership [ACP09, BPV12b]. Basically, SPHFs are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a language L. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where hk, the hashing key, can be seen as the private key and hp, the projection key, as the public key. On a word $W \in \mathrm{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ with the hashing key and $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, W, w)$ with the projection key only but also a witness w that $W \in \mathrm{L}$. Of course, if $W \notin \mathrm{L}$, such a witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ is independent of hp. As a consequence, even knowing hp, one cannot guess $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$.

One-Round PAKE in the BPR Model.

Gennaro and Lindell [GL03] (GL) extended the initial definition of smooth projective hash functions for an application to PAKE. Their approach has thereafter been adapted to the *Universal Composability* (UC) framework by Canetti *et al.* [CHK+05b], but for static corruptions only. It has been improved by Abdalla, Chevalier and Pointcheval [ACP09] to resist to adaptive adversaries. But the 3-flow KOY protocol remains the most efficient protocol BPR-secure under the DDH assumption.

More recently, the ultimate step for PAKE has been achieved by Katz and Vaikuntanathan [KV11] (KV), who proposed a *practical* one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own passwords only. More precisely, each flow just consists of an IND-CCA ciphertext of the password and an SPHF projection key for the correctness of the partner's ciphertext (the word is the ciphertext and the witness consists of the random coins of the encryption). The shared secret key is eventually the product of the two hash values, as in the KOY and GL protocols. Because of the simultaneous flows, one flow cannot explicitly depend on the partner's flow, which makes impossible the use of the Gennaro and Lindell SPHF (later named GL-SPHF), in which the projection key depends on the word (the ciphertext here). On the other hand, the adversary can wait for the player to send his flow first, and then adapt its message, which requires stronger security notions than the initial Cramer and Shoup SPHF (later named CS-SPHF), in which the smoothness does not hold anymore if the word is generated after having seen the projection key. This led Katz and Vaikuntanathan to provide a new definition for SPHF (later named KV-SPHF), where the pro-

jection key depends on the hashing key only, and the smoothness holds even if the word is chosen after having seen the projection key. Variations between CS-SPHF, GL-SPHF and KV-SPHF are in the way one computes the projection key hp from the hashing key hk and the word W, but also in the smoothness property, according to the freedom the adversary has to choose W, when trying to distinguish the hash value from a random value. As a side note, while CS-SPHF is close to the initial definition, useful for converting an IND-CPA encryption scheme to IND-CCA, GL-SPHFs and KV-SPHFs did prove quite useful too: we will use KV-SPHFs for our one-round PAKE protocols and a GL-SPHF for the blind signature scheme.

As just explained, the strongest definition of SPHF, which gives a lot of freedom to the adversary, is the recent KV-SPHF. However, previous SPHFs known on Cramer-Shoup ciphertexts were GL-SPHFs only. For their one-round PAKE, Katz and Vaikuntanathan did not manage to construct such a KV-SPHF for an efficient IND-CCA encryption scheme. They then suggested to use the Naor and Yung approach [NY90], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [Sah99]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [GS08] proofs in bilinear groups and the linear encryption [BBS04] which leads to a PAKE secure under the DLin assumption with a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, the two players have to send 70 group elements each, which is far more costly than the KOY protocol, but it is one-round only.

More recent results on SS-NIZK proofs or IND-CCA encryption schemes, in the discrete logarithm setting, improved on that: Libert and Yung [LY12] proposed a more efficient SS-NIZK proof of plaintext equality in the Naor-Yung-type cryptosystem with ElGamal-like encryption. The proof can be reduced from 60 to 22 group elements and the communication complexity of the resulting PAKE is decreased to 32 group elements per user. Jutla and Roy [JR12] proposed relatively-sound NIZK proofs as an efficient alternative to SS-NIZK proofs to build new publicly-verifiable IND-CCA encryption schemes. They can then decrease the PAKE communication complexity to 20 group elements per user. In any case, one can remark that all one-round PAKE schemes require pairing computations.

Language-Authenticated Key Exchange.

A generalization of AKE protocols has been recently proposed, so-called *Language-Authenticated Key Exchange* (LAKE) [BBC+13a]: it allows two users, Alice and Bob, each owning a word in a specific language, to agree on a shared high entropy secret if each user knows a word in the language the other thinks about. More precisely, they first both agree on public parameters pub, Bob will think about *priv* for his expected Alice's value of priv, Alice will do the same with *priv*′ for Bob's private value priv′; eventually, if *priv* = priv and *priv*′ = priv′, and if they both know words in the appropriate languages, then the key agreement will succeed. In case of failure, no information should leak to the players about the reason of failure, except that the inputs did not satisfy the relations, or the languages were not consistent. Eavesdroppers do not even learn the outcome.

This formalism encompasses PAKE, and their first construction follows the GL approach for PAKE: each player commits to the private values (his own value priv, and his expected partner's value *priv*′) as well as his own word, and projection keys are sent to compute random values that will be the same if and only if everything is consistent. To achieve one-round LAKE, one also needs KV-SPHF on ciphertexts for plaintext-equality tests (equality of the private values and expected private values) and for language-membership.

Achievements.

Our main contribution is the description of an instantiation of KV-SPHF on Cramer-Shoup ciphertexts, and thus the first KV-SPHF on an efficient IND-CCA encryption scheme. We thereafter

use it within the above KV framework for one-round PAKE [KV11], in the BPR security model. Our scheme just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key). This has to be compared with the 20 group elements, or more, in the best constructions discussed above, which all need pairing-friendly groups and pairing computations, or with the KOY protocol that has a similar complexity but with three sequential flows.

We also present the first GL-SPHFs/KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. Those SPHFs are thus quite efficient. They lead to two applications. First, our new KV-SPHFs enable several efficient instantiations of one-round *Language-Authenticated Key-Exchange* (LAKE) protocols [BBC+13a]. Our above one-round PAKE scheme is actually a particular case of a more general one-round LAKE scheme, for which we provide a BPR-like security model and a security proof. Our general constructions also cover Credential-Authenticated Key Exchange [CCGS10]. Second, thanks to a new GL-SPHF, we improve on the *blind signature* scheme presented in [BPV12b], from $5\ell + 6$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$ to $3\ell + 7$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$, for an ℓ-bit message to be blindly signed with a Waters signature [Wat05]. Our protocol is round-optimal, since it consists of two flows, and leads to a classical short Waters signature.

As a side contribution, we introduce a new generic framework to construct SPHFs aiming at making easier the construction and the proof of SPHFs on complex languages. Using this framework, we were able to construct SPHFs for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language.

Outline of the Paper.

In Section I.2, we first revisit the different definitions for SPHFs proposed in [CS02,GL03,KV11], denoted respectively CS-SPHFs, GL-SPHFs and KV-SPHFs. While CS-SPHF was the initial definition useful for converting an IND-CPA encryption scheme to IND-CCA, GL-SPHFs and KV-SPHFs did prove quite useful too: we will use a KV-SPHF for our PAKE/LAKE application and a GL-SPHF for the blind signature. In Section I.2.4, we introduce our main contribution, the construction of a KV-SPHF on Cramer-Shoup ciphertexts. This KV-SPHF leads to the construction of our efficient one-round PAKE in Section I.2.5. In Section I.3, we present a simplified version of our generic framework (fully described in Appendix I.D). We then show our efficient SPHFs on multi-exponentiation equations and on bit encryption, without pairings, in Section I.4. Finally, in Section I.5, we introduce our two other constructions based on these SPHFs: our one-round LAKE and our blind signature scheme.

I.2 New SPHF on Cramer-Shoup Ciphertexts and PAKE

In this section, we first recall the definitions of SPHFs and present our classification based on the dependence between words and keys. According to this classification, there are three types of SPHFs: the (almost) initial Cramer and Shoup [CS02] type (CS-SPHF) introduced for enhancing an IND-CPA encryption scheme to IND-CCA, the Gennaro and Lindell [GL03] type (GL-SPHF) introduced for PAKE, and the Katz and Vaikuntanathan [KV11] type (KV-SPHF) introduced for one-round PAKE.

Then, after a quick review on the Cramer-Shoup encryption scheme, we introduce our new KV-SPHF on Cramer-Shoup ciphertexts which immediately leads to a quite efficient instantiation of the Katz and Vaikuntanathan one-round PAKE [KV11], secure in the BPR model.

I.2.1 General Definition of SPHFs

Let us consider a language $\mathrm{L} \subseteq \mathcal{S}et$, and some global parameters for the SPHF, assumed to be in the common random string (CRS). The SPHF system for the language L is defined by four algorithms:

- $\mathsf{HashKG}(\mathrm{L})$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C)$ derives the projection key hp, possibly depending on the word C;
- $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)$ outputs the hash value of the word C from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$ outputs the hash value of the word C from the projection key hp, and the witness w that $C \in \mathrm{L}$.

The *correctness* of the SPHF assures that if $C \in \mathrm{L}$ with w a witness of this membership, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin \mathrm{L}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp. For that, we use the classical notion of statistical distance recalled in I.A.2.

I.2.2 Smoothness Adaptivity and Key Word-Dependence

This paper will exploit the very strong notion KV-SPHF. Informally, while the GL-SPHF definition allows the projection key hp to depend on the word C, the KV-SPHF definition prevents the projection key hp from depending on C, as in the original CS-SPHF definition. In addition, the smoothness should hold even if C is chosen as an arbitrary function of hp. This models the fact the adversary can see hp before deciding which word C it is interested in. More formal definitions follow, where we denote Π the range of the hash function.

CS-SPHF.

This is almost[1] the initial definition of SPHF, where the projection key hp does not depend on the word C (word-independent key), but the word C cannot be chosen after having seen hp for breaking the smoothness (non-adaptive smoothness). More formally, a CS-SPHF is ε-smooth if ProjKG does not use its input C and if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

GL-SPHF.

This is a relaxation, where the projection key hp can depend on the word C (word-dependent key). More formally, a GL-SPHF is ε-smooth if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \xleftarrow{\$} \Pi\}.$$

[1] In the initial definition, the smoothness was defined for a word C randomly chosen from $\mathcal{S}et \backslash \mathrm{L}$, and not necessarily for any such word.

KV-SPHF.

This is the strongest SPHF, in which the projection key hp does not depend on the word C (word-independent key) and the smoothness holds even if C depends on hp (adaptive smoothness). More formally, a KV-SPHF is ε-smooth if ProjKG does not use its input C and, for any function f onto $\mathcal{S}et\backslash\mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, f(\mathsf{hp}))\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

Remark I.2.1 One can see that a perfectly smooth (*i.e.*, 0-smooth) CS-SPHF is also a perfectly smooth KV-SPHF, since each value H has exactly the same probability to appear, and so adaptively choosing C does not increase the above statistical distance. However, as soon as a weak word C can bias the distribution, f can exploit it.

I.2.3 SPHFs on Languages of Ciphertexts

We could cover languages as general as those proposed in [BBC+13a], but for the sake of clarity, and since the main applications need some particular cases only, we focus on SPHFs for languages of ciphertexts, whose corresponding plaintexts verify some relations. We denote these languages $\textsc{Lof}\mathrm{C}_{\mathsf{full\text{-}aux}}$.

The parameter full-aux will parse in two parts $(\mathsf{crs}, \mathsf{aux})$: the public part crs, known in advance, and the private part aux, possibly chosen later. More concretely, crs represents the public values: it will define the encryption scheme (and will thus contain the global parameters and the public key of the encryption scheme) with the global format of both the tuple to be encrypted and the relations it should satisfy, and possibly additional public coefficients; while aux represents the private values: it will specify the relations, with more coefficients or constants that will remain private, and thus implicitly known by the sender and the receiver (as the expected password, for example, in PAKE protocols).

To keep aux secret, hp should not leak any information about it. We will thus restrict HashKG and ProjKG not to use the parameter aux, but just crs. This is a stronger restriction than required for our purpose, since one can use aux without leaking any information about it. But we already have quite efficient instantiations, and it makes everything much simpler to present.

I.2.4 SPHFs on Cramer-Shoup Ciphertexts

Labeled Cramer-Shoup Encryption Scheme (CS).

The CS labeled encryption scheme is recalled in I.A.3. We briefly review it here. We combine all the public information in the encryption key. We thus have a group $\mathbb{G}$ of prime order p, with two independent generators $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, a hash function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$ from a collision-resistant hash function family onto $\mathbb{Z}_p^*$, and a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. From 5 scalars $(x_1, x_2, y_1, y_2, z) \xleftarrow{\$} {\mathbb{Z}_p}^5$, one also sets $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. The encryption key is $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$, while the decryption key is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$. For a message $m \in \{0,1\}^n$, with $M = \mathcal{G}(m) \in \mathbb{G}$, the labeled Cramer-Shoup ciphertext is:

$$C \stackrel{\mathsf{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, M; r) \stackrel{\mathsf{def}}{=} (\mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r),$$

with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e) \in \mathbb{Z}_p^*$. If one wants to encrypt a vector of group elements $(M_1, \ldots, M_n)$, all at once in a non-malleable way, one computes all the individual ciphertexts with a common $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$ for $v_1, \ldots, v_n$. Hence, everything done on tuples of ciphertexts will work on ciphertexts of vectors. In addition, the Cramer-Shoup labeled encryption scheme on vectors is IND-CCA under the DDH assumption.

The (known) GL-SPHF for CS.

Gennaro and Lindell [GL03] proposed an SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $C = (\mathbf{u} = (u_1, u_2) = (g_1^r, g_2^r), e = \mathcal{G}(m) \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^\eta g_2^\theta h^\mu (cd^\xi)^\nu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language $\text{LoF}\text{C}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$, where $\mathsf{crs} = \mathsf{ek}$ is public but $\mathsf{aux} = m$ is kept secret:

$$H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) \stackrel{\text{def}}{=} u_1^\eta u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu$$
$$= \mathsf{hp}^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) \stackrel{\text{def}}{=} H'.$$

A (new) KV-SPHF for CS.

We give here the description of the first known KV-SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$; the associated projection key is the pair $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\mu c^\nu, \mathsf{hp}_2 = g_1^{\eta_2} d^\nu) \in \mathbb{G}^2$. Then one can compute the hash value in two different ways, for the language $\text{LoF}\text{C}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$ under ek:

$$H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) \stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu$$
$$= (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'.$$

Theorem I.2.2 The above SPHF is a perfectly smooth (*i.e.*, 0-smooth) KV-SPHF.

The proof can be found in Section I.D.3 as an illustration of our new framework.

I.2.5 An Efficient One-Round PAKE

Review of the Katz and Vaikuntanathan's PAKE.

As explained earlier, Katz and Vaikuntanathan recently proposed a one-round PAKE in [KV11]. Their general framework follows Gennaro and Lindell [GL03] approach: each player sends an encryption of the password, and then uses an SPHF on the partner's ciphertext to check whether it actually contains the same password. The two hash values are multiplied to produce the session key. If the encrypted passwords are the same, the different ways to compute the hash values (Hash and ProjHash) give the same results. If the passwords differ, the smoothness makes the values computed by each player independent. To this aim, the authors need an SPHF on a labeled IND-CCA encryption scheme. To allow a SPHF-based PAKE scheme to be one-round, the ciphertext and the SPHF projection key for verifying the correctness of the partner's ciphertext should be sent together, before having seen the partner's ciphertext: the projection key should be independent of the ciphertext. In addition, the adversary can wait until it receives the partner's projection key before generating the ciphertext, and thus a stronger smoothness is required. This is exactly why we need a KV-SPHF in this one-round PAKE framework.

Our Construction.

Our KV-SPHF on Cramer-Shoup ciphertexts can be used in the Katz and Vaikuntanathan framework for PAKE [KV11]. It leads to the most efficient PAKE known so far, and it is *one-round*. Each user indeed only sends 6 elements of $\mathbb{G}$ (see Figure I.1), instead of 70 elements of $\mathbb{G}$ for the Katz and Vaikuntanathan's instantiation using a Groth-Sahai SS-NIZK [GS08], or 20 group elements for the Jutla and Roy's [JR12] improvement using a relatively-sound NIZK.

- Players U and U' both use $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$;
- U, with password pw, chooses $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$,
  computes $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu})$, sets $\ell = (U, U', \mathsf{hp})$,
  and generates $C = (\mathbf{u} = (g_1^r, g_2^r), e = \mathcal{G}(\mathsf{pw}) \cdot h^r, v = (cd^{\xi})^r)$ with r a random scalar in $\mathbb{Z}_p$ and $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
  U sends $\mathsf{hp} \in \mathbb{G}^2$ and $C \in \mathbb{G}^4$ to U';
- Upon receiving $\mathsf{hp}' = (\mathsf{hp}'_1, \mathsf{hp}'_2) \in \mathbb{G}^2$ and $C' = (\mathbf{u}' = (u'_1, u'_2), e', v') \in \mathbb{G}^4$ from U', U sets $\ell' = (U', U, \mathsf{hp}')$ and $\xi' = \mathfrak{H}_K(\ell', \mathbf{u}', e')$ and computes
$$\mathsf{sk}_U = {u'_1}^{(\eta_1 + \xi' \eta_2)} {u'_2}^{\theta} (e'/\mathcal{G}(\mathsf{pw}))^{\mu} {v'}^{\nu} \cdot (\mathsf{hp}'_1 {\mathsf{hp}'_2}^{\xi})^r.$$

Figure I.1: One-Round PAKE based on DDH

The formal security result follows from the Theorem I.5.2 in Section I.5.1. We want to insist that our construction does not need pairing-friendly groups, and the plain DDH assumption is enough, whereas the recent constructions made heavy use of pairing-based proofs *à la* Groth-Sahai. Under the DLin assumption (which is a weaker assumption in any group), still without requiring pairing-friendly groups, our construction would make each user to send 9 group elements only.

I.3 Generic Framework for SPHFs

I.3.1 Introduction

In I.D, we propose a formal framework for SPHFs using a new notion of graded rings, derived from [GGH13a]. It enables to deal with cyclic groups, bilinear groups (with symmetric or asymmetric pairings), or even groups with multi-linear maps. In particular, it helps to construct concrete SPHFs for quadratic pairing equations over ciphertexts, which enable to construct efficient LAKE [BBC+13a] for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language (see Section I.5.1).

However, we focus here on cyclic groups, with the basic intuition only, and provide some illustrations. While we keep the usual multiplicative notation for the cyclic group $\mathbb{G}$, we use an extended notation: $r \odot u = u \odot r = u^r$, for $r \in \mathbb{Z}_p$ and $u \in \mathbb{G}$, and $u \oplus v = u \cdot v$, for $u, v \in \mathbb{G}$. Basically, $\oplus$ and $\odot$ correspond to the addition and the multiplication in the exponents, that are thus both commutative. We then extend this notation in a natural way when working on vectors and matrices.

Our goal is to deal with languages of ciphertexts $\text{LoFC}_{\mathsf{full\text{-}aux}}$: we assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \text{LoFC}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$.

I.3.2 Language Representation

For a language $\mathrm{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : \mathcal{S}et \mapsto \mathbb{G}^{k \times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathbb{G}^{1 \times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \vec{\lambda} \in \mathbb{Z}_p^{1 \times k}$ such that $\Theta_{\mathsf{aux}}(C) = \vec{\lambda} \odot \Gamma(C))$. In other words, we assume that $C \in \mathrm{L}_{\mathsf{aux}}$, if and only if, $\Theta_{\mathsf{aux}}(C)$ is a linear combination of (the exponents in) the rows of some matrix $\Gamma(C)$. For a KV-SPHF, Γ is supposed to be a constant function (independent of the word C). Otherwise, one gets a GL-SPHF.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute the above *linear* combination $\vec{\lambda}$. This may seem a quite strong requirement

but this is actually verified by very expressive languages over ciphertexts such as ElGamal, Cramer-Shoup and variants.

We briefly illustrate it on our KV-SPHF on CS: $C = (u_1 = g_1^r, u_2 = g_2^r, e = M \cdot h^r, v = (cd^\xi)^r)$, with $k = 2$, $\mathsf{aux} = M$ and $n = 5$:

$$\Gamma = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \qquad \vec{\lambda} = (r, r\xi) \qquad \begin{array}{rcl} \vec{\lambda} \odot \Gamma & = & (g_1^r, g_1^{r\xi}, g_2^r, h^r, (cd^\xi)^r) \\ \Theta_M(C) & = & (u_1, u_1^\xi, u_2, e/M, v). \end{array}$$

Essentially, one tries to make the first columns of $\Gamma(C)$ and the first components of $\Theta_{\mathsf{aux}}(C)$ to completely determine $\vec{\lambda}$. In our illustration, the first two columns with $u_1 = g_1^r$ and $u_1^\xi = g_1^{r\xi}$ really imply $\vec{\lambda} = (r, r\xi)$, and the three last columns help to check the language membership: we want $u_2 = g_2^r$, $e/M = h^r$, and $v = (cd^\xi)^r$, with the same r as for u_1.

I.3.3 Smooth Projective Hash Function

With the above notations, the hashing key is a vector $\mathsf{hk} = \vec{\alpha} = (\alpha_1, \ldots, \alpha_n)^\intercal \xleftarrow{\$} \mathbb{Z}_p^n$, while the projection key is, for a word C, $\mathsf{hp} = \vec{\gamma}(C) = \Gamma(C) \odot \vec{\alpha} \in \mathbb{G}^k$ (if Γ depends on C, this leads to a GL-SPHF, otherwise, one gets a KV-SPHF). Then, the hash value is:

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\text{def}}{=} \Theta_{\mathsf{aux}}(C) \odot \vec{\alpha} = \vec{\lambda} \odot \vec{\gamma}(C) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w).$$

Our above Γ, $\vec{\lambda}$, and Θ_M immediately lead to our KV-SPHF on CS from the Section I.2.4: with $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$, the product with Γ leads to: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\mu c^\nu, \mathsf{hp}_2 = g_1^{\eta_2} d^\nu) \in \mathbb{G}^2$, and

$$\begin{aligned} H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'. \end{aligned}$$

The generic framework detailed in I.D also contains a security analysis that proves the above generic SPHF is perfectly smooth: Intuitively, for a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \vec{\gamma}(C) = \Gamma(C) \odot \vec{\alpha}$, the vector $\Theta_{\mathsf{aux}}(C)$ is not in the linear span of $\Gamma(C)$, and thus $H = \Theta_{\mathsf{aux}}(C) \odot \vec{\alpha}$ is independent from $\Gamma(C) \odot \vec{\alpha} = \mathsf{hp}$.

I.4 Concrete Constructions of SPHFs

In this section, we illustrate more our generic framework, by constructing more evolved SPHFs without pairings. More complex constructions of SPHFs, namely for any language handled by the Groth-Sahai NIZK proofs, are detailed in I.D.

I.4.1 KV-SPHF for Linear Multi-Exponentiation Equations

We present several instantiations of KV-SPHFs, in order to illustrate our framework, but also to show that our one-round PAKE protocol from Section I.2.5 can be extended to one-round LAKE [BBC+13a]. In PAKE/LAKE, we use SPHFs to prove that the plaintexts associated with some ElGamal-like ciphertexts verify some relations. The communication complexity of these protocols depends on the ciphertexts size and of the projection keys size. We first focus on ElGamal ciphertexts, and then explain how to handle Cramer-Shoup ciphertexts.

Notations.

We work in a group $\mathbb{G}$ of prime order p, generated by g, in which we assume the DDH assumption to hold. We define ElGamal encryption scheme with encryption key $\mathsf{ek} = (g, h = g^x)$. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i)$, for $X_1, \ldots, X_{n_1} \in \mathbb{G}$, and $C_{2,j} = (u_{2,j} = g^{r_{2,j}}, e_{2,j} = h^{r_{2,j}} \cdot g^{y_j})$, for $y_1, \ldots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{rcl} \mathsf{crs} & = & (p, \mathbb{G}, \mathsf{ek}, A_1, \ldots, A_{n_2}) \\ \mathsf{aux} & = & (a_1, \ldots, a_{n_1}, B) \in \mathbb{Z}_p^{n_1} \times \mathbb{G}. \end{array} \tag{I.1}$$

We insist that, here, the elements $(A_1, \ldots, A_{n_2}) \in \mathbb{G}^{n_2}$ are known in advance, contrarily to equation (I.2) in I.D.4, where they are in aux and make the SPHF to use pairings.

In the following, i and j will always range from 1 to n_1 and from 1 to n_2 respectively in all the products $\prod_i$, $\prod_j$ and tuples $(\cdot)_i$, $(\cdot)_j$. We can define the following elements, and namely the $(2n_2+1, 2n_2+2)$-matrix Γ that uses the knowledge of the elements $(A_j)_j$:

$$\Gamma = \left(\begin{array}{c|ccc|ccc|c} g & 1 & \ldots & 1 & 1 & \ldots & 1 & h \\ \hline 1 & g & & 1 & h & & 1 & 1 \\ \vdots & & \ddots & & & \ddots & & \vdots \\ 1 & 1 & & g & 1 & & h & 1 \\ \hline 1 & & 1 & & g & & 1 & A_1^{-1} \\ \vdots & & & & & \ddots & & \vdots \\ 1 & & & & 1 & & g & A_{n_2}^{-1} \end{array}\right)$$

and

$$\begin{array}{rcl} \Theta_{\mathsf{aux}}(\vec{C}) & = & \left(\prod_i u_{1,i}^{a_i}, (u_{2,j})_j, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \\ \vec{\lambda} & = & (\sum_i a_i r_{1,i}, (r_{2,j})_j, (y_j)_j) \\ \vec{\lambda} \odot \Gamma & = & \left(g^{\sum_i a_i r_{1,i}}, (g^{r_{2,j}})_j, (h^{r_{2,j}} \cdot g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \prod_j A_j^{y_j}\right). \end{array}$$

We recall that in the matrix, 1 is the neutral element in $\mathbb{G}$ and can thus be ignored. When one considers the discrete logarithms, they become 0, and thus the matrix is triangular. The three diagonal blocks impose the value of $\vec{\lambda}$, and the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\vec{C})$ is $\prod_i e_{1,i}^{a_i}/B = h^{\sum_i a_i r_{1,i}} \cdot \prod_i X_i^{a_i}/B$, which is equal to the last component of $\vec{\lambda} \odot \Gamma = h^{\sum_i a_i r_{1,i}} / \prod_j A_j^{y_j}$, if and only if the relation (I.1) is satisfied. It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g^\eta h^\nu$, $(\mathsf{hp}_{2,j} = g^{\theta_j} h^{\mu_j})_j$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\theta_j)_j, (\mu_j)_j, \nu)$:

$$H = \prod_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod_j (u_{2,j}^{\theta_j} e_{2,j}^{\mu_j})/B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \prod_j (\mathsf{hp}_{2,j}^{r_{2,j}} \cdot \mathsf{hp}_{3,j}^{y_j}) = H'.$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 4n_2 + 1$ elements from $\mathbb{G}$.

Ciphertexts with Randomness Reuse.

A first improvement consists in using multiple independent encryption keys for encrypting the y_j's: $\mathsf{ek}_{2,j} = (g, h_{2,j} = g^{x_{2,j}})$, for $j = 1, \ldots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g^{y_j})_j)$, for

$(y_j)_j \in \mathbb{Z}_p^{n_2}$, still satisfying the same relation (I.1). This improves on the length of the ciphertexts, from $2n_1+2n_2$ group elements in $\mathbb{G}$ to $2n_1+n_2+1$. The matrix Γ can then be compressed into:

$$\Gamma = \left(\begin{array}{c|c|ccc|c} g & 1 & 1 & \dots & 1 & h \\ \hline 1 & g & h_{2,1} & \dots & h_{2,n_2} & 1 \\ \hline 1 & 1 & g & & \mathbf{1} & A_1^{-1} \\ \vdots & \vdots & & \ddots & & \vdots \\ 1 & 1 & \mathbf{1} & & g & A_{n_2}^{-1} \end{array}\right)$$

and

$$\begin{aligned} \Theta_{\mathsf{aux}}(\vec{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, u_2, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \\ \vec{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, r_2, (y_j)_j) \\ \vec{\lambda} \odot \Gamma &= (g^{\sum_i a_i r_{1,i}}, g^{r_2}, (h_{2,j}^{r_2} g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \textstyle\prod_j A_j^{y_j}) \end{aligned}$$

where again, because of the diagonal blocks in Γ, $\vec{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\vec{C})$. The last component of $\Theta_{\mathsf{aux}}(\vec{C})$ is then $\prod_i e_{1,i}^{a_i}/B = \prod_i h^{a_i r_{1,i}} X_i^{a_i}/B$ and thus equal to the last component of $\vec{\lambda} \odot \Gamma$, multiplied by $\prod_i X_i^{a_i} \cdot \prod_j A_j^{y_j}/B$ that is equal to 1 if and only if the relation (I.1) is satisfied. It thus leads to the following KV-SPHF, with ($\mathsf{hp}_1 = g^\eta h^\nu$, $\mathsf{hp}_2 = g^\theta \cdot \prod_j h_{2,j}^{\mu_j}$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, \theta, (\mu_j)_j, \nu)$):

$$H = \prod_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod_j e_{2,j}^{\mu_j} \cdot u_2^\eta / B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \mathsf{hp}_2^{r_2} \cdot \prod_j \mathsf{hp}_{3,j}^{y_j} = H'.$$

Globally, the ciphertexts and the projection keys consist of $2n_1+2n_2+3$ elements from $\mathbb{G}$. This has to be compared with $2n_1+4n_2+1$ elements from $\mathbb{G}$ in the previous construction.

Moving all the constant values from aux to crs.

In some cases, all the constant values, A_j and a_i can be known in advance and public. The matrix Γ can then exploit their knowledge. We apply the randomness-reuse technique for the whole ciphertext, for both $(X_i)_i$ and $(y_j)_j$, with independent encryption keys $(h_{1,i})_i$ and $(h_{2,j})_j$ in $\mathbb{G}$. A unique random r produces $u = g^r$, and $(e_{1,i})_i$ and $(e_{2,j})_j$. This reduces the length of the ciphertext to n_1+n_2+1 group elements in $\mathbb{G}$, but also the size of the matrix Γ:

$$\Gamma = \left(\begin{array}{c|ccc|c} g & h_{2,1} & \dots & h_{2,n_2} & \prod_i h_{1,i}^{a_i} \\ \hline 1 & g & & \mathbf{1} & A_1^{-1} \\ \vdots & & \ddots & & \vdots \\ 1 & \mathbf{1} & & g & A_{n_2}^{-1} \end{array}\right) \qquad \begin{aligned} \Theta_{\mathsf{aux}}(\vec{C}) &= \left(u, (e_{2,j})_j, \textstyle\prod_i e_{1,i}^{a_i}/B\right) \\ \vec{\lambda} &= (r, (y_j)_j) \\ \vec{\lambda} \odot \Gamma &= (g^r, (h_{2,j}^r g^{y_j})_j, \textstyle\prod_i h_{1,i}^{a_i r} / \prod_j A_j^{y_j}) \end{aligned}$$

Projection keys become more compact, with only n_2+1 group elements in $\mathbb{G}$: $\mathsf{hp}_1 = g_1^\eta \cdot \prod_j h_{2,j}^{\mu_j} \cdot (\prod_i h_{1,i}^{a_i})^\nu$, and $(\mathsf{hp}_{2,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\mu_j)_j, \nu)$: $H = u^\eta \cdot \prod_i e_{1,i}^{\nu a_i} \cdot \prod_j e_{2,j}^{\mu_j} / B^\nu = \mathsf{hp}_1^r \cdot \prod_j \mathsf{hp}_{2,j}^{y_j} = H'$. Globally, the ciphertexts and the projection keys consist of n_1+2n_2+2 elements from $\mathbb{G}$.

I.4.2 From ElGamal to Cramer-Shoup Encryption

In order to move from ElGamal ciphertexts to Cramer-Shoup ciphertexts, if one already has Γ, Θ_{aux} and $\vec{\Lambda}$, to guarantee that the ElGamal plaintexts satisfy a relation, one simply has to make a bigger matrix, diagonal per blocks, with blocks Γ and smallers $(\Gamma_k)_k$ for every ciphertext $(u_k, u'_k, e_k, v_k)_k$, where

$$\Gamma_k = \begin{pmatrix} g & 1 & g' & c \\ 1 & g & 1 & d \end{pmatrix} \qquad \vec{\lambda_k} = (r_k, r_k \xi_k) \qquad \begin{aligned} \Theta_M(C_k) &= (u_k, u_k^{\xi_k}, u'_k, v_k) \\ \vec{\lambda}_k \odot \Gamma_k &= (g^{r_k}, g^{r_k \xi_k}, g'^{r_k}, (c d^{\xi_k})^{r_k}) \end{aligned}$$

The initial matrix Γ guarantees the relations on the ElGamal pairs (u_k, e_k), and the matrices Γ_k add the internal relations on the Cramer-Shoup ciphertexts. In the worst case, hk is increased by $4n$ scalars and hp by $2n$ group elements, for n ciphertexts. But some more compact matrices can be obtained in many cases, with much shorter hashing and projection keys, by merging some lines or columns in the global matrix. But this is a case by case optimization.

I.4.3 Generalizations

The SPHF constructions from this section are all done without requiring any pairing, but are still KV-SPHF, allowing us to handle non-quadratic multi-exponentiation equations without pairings. To further extend our formalism, we describe in the next section a concrete application to blind signatures (while with a GL-SPHF), and we present more languages in I.D.4.

However, as above for Cramer-Shoup ciphertexts, if one wants to satisfy several equations at a time, one just has to first consider them independently and to make a global matrix with each sub-language-matrix in a block on the diagonal. The hashing keys and the projection keys are then concatenated, and the hash values are simply multiplied. Optimizations can be possible, as shown in I.C for the SPHF involved in the blind signature.

I.4.4 GL-SPHF on Bit Encryption

As shown in I.D, our general framework allows to construct KV-SPHFs for any language handled by the Groth-Sahai NIZK proofs. But, while these KV-SPHFs encompass the language of ciphertexts encrypting a bit, they require pairing evaluations. We show here a more efficient GL-SPHF for bit encryption, which does not need pairings.

Let us consider an ElGamal ciphertext $C = (u = g^r, e = h^r g^y)$, in which one wants to prove that $y \in \{0, 1\}$. We can define the following matrix that depends on C, hence a GL-SPHF:

$$\Gamma(C) = \begin{pmatrix} g & h & 1 & 1 \\ 1 & g & u & e/g \\ 1 & 1 & g & h \end{pmatrix} \qquad \begin{array}{rcl} \Theta_{\mathsf{aux}}(C) & = & (u, e, 1, 1) \qquad \vec{\lambda} = (r, y, -ry) \\ \vec{\lambda} \odot \Gamma(C) & = & (g^r, h^r g^y, (u/g^r)^y, (e/gh^r)^y) \end{array}$$

Because of the triangular block in $\Gamma(C)$, one sees that $\Theta_{\mathsf{aux}}(C) = \vec{\lambda} \odot \Gamma(C)$ if and only if $g^{y(y-1)} = 1$, and thus that $y \in \{0, 1\}$. With $\mathsf{hp}_1 = g^\nu h^\theta$, $\mathsf{hp}_2 = g^\theta u^\eta (e/g)^\lambda$, and $\mathsf{hp}_3 = g^\eta h^\lambda$, for $\mathsf{hk} = (\nu, \theta, \eta, \lambda)$: $H = u^\nu e^\theta = \mathsf{hp}_1^r \cdot \mathsf{hp}_2^y / \mathsf{hp}_3^{ry} = H'$.

I.5 More Applications of SPHFs

I.5.1 One-Round LAKE

Since we have shown that our framework allows to design KV-SPHFs for complex languages, we extend our PAKE protocol to LAKE [BBC+13a]. To this aim, we provide a new security model, inspired from BPR [BPR00] and a complete security proof, which implies the security of our PAKE protocol from Section I.2.5.

Review of Language-Authenticated Key Exchange.

LAKE is a general framework [BBC+13a] that generalizes AKE primitives: each player U owns a word W in a certain language $\mathcal{L}$ and expects the other player to own a word W' in a language $\mathcal{L}'$. If everything is compatible (*i.e.*, the languages are the expected languages and the words are indeed in the appropriate languages), the players compute a common high-entropy secret key, otherwise they learn nothing about the partner's values. In any case, external eavesdroppers do not learn anything, even not the outcome of the protocol: did it succeed or not?

More precisely, we assume the two players have initially agreed on a common public part pub for the languages, but then they secretly parametrize the languages with the private parts priv: $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ is the language they want to use, and $\mathcal{L}_{\mathsf{pub},priv'}$ is the language they assume the other player will use. In addition, each player owns a word W in his language. We will thus have to use SPHFs on ciphertexts on W, priv and $priv'$, with a common $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$ and aux with the private parameters. For simple languages, this encompasses PAKE and Verifier-based PAKE. We refer to [BBC+13a] for more applications of LAKE.

A New Security Model for LAKE.

The first security model for LAKE [BBC+13a] has been given in the UC framework [Can01], as an extension of the UC security for PAKE [CHK+05b]. In this paper, we propose an extension of the PAKE security model presented by Bellare, Pointcheval, and Rogaway [BPR00] model for LAKE: the adversary $\mathcal{A}$ plays a find-then-guess game against n players $(P_i)_{i=1,\ldots,n}$. It has access to several instances Π_U^s for each player $U \in \{P_i\}$ and can activate them (in order to model concurrent executions) via several queries: Execute-queries model passive eavesdroppings; Send-queries model active attacks; Reveal-queries model a possible bad later use of the session key; the Test-query models the secrecy of the session key. The latter query has to be asked to a *fresh* instance (which basically means that the session key is not trivially known to the adversary) and models the fact that the session key should look random for an outsider adversary.

Our extension actually differs from the original PAKE security model [BPR00] when defining the quality of an adversary. The goal of an adversary is to distinguish the answer of the Test-query on a fresh instance: a trivial attack is the so-called on-line dictionary attack which consists in trying all the possibilities when interacting with a target player. For PAKE schemes, the advantage of such an attack is q_s/N, where q_s is the number of Send-queries and N the number of possible passwords. A secure PAKE scheme should guarantee this is the best attack, or equivalently that the advantage of any adversary is bounded by $q_s \times 2^{-m}$, where m is the min-entropy of the password distribution. In our extension, for LAKE, the trivial attack consists in trying all the possibilities for $\mathsf{priv}, priv'$ with a word W in $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

Definition I.5.1 [Security for LAKE] A LAKE protocol is claimed (t, ε)-secure if the advantage of any adversary running in time t is bounded by $q_s \times 2^{-m} \times \mathsf{Succ}^{\mathcal{L}}(t) + \varepsilon$, where m is the min-entropy of the pair $(\mathsf{priv}, \mathsf{priv}')$, and $\mathsf{Succ}^{\mathcal{L}}(t)$ is the maximal success an adversary can get in finding a word in any $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ within time t.

Note that the min-entropy of the pair $(\mathsf{priv}, \mathsf{priv}')$ might be conditioned to the public information from the context.

Our Instantiation.

Using the same approach as Katz and Vaikuntanathan for their one-round PAKE [KV11], one can design the scheme proposed on Figure I.2, in which both users U and U' use the encryption key ek and the public part pub. This defines $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$. When running the protocol, U owns a word W for a private part priv, and thinks about a private part $priv'$ for U', while U' owns a word W' for a private part priv', and thinks about a private $priv$ for U.

This gives a concrete instantiation of one-round LAKE as soon as one can design a KV-SPHF on the language

$$\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')} = \{(\ell, C) \mid \exists r, \exists W,\ C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r) \text{ and } W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}\}.$$

More precisely, each player encrypts $(\mathsf{priv}, priv', W)$ as a vector, which thus leads to $C = (C_1, C_2, C_3)$. We then use the combination of three SPHFs: two on equality-test for the plaintexts priv (for C_1) and $priv'$ (for C_2), and one on $\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),\mathsf{priv}}$ for the ciphertext C_3 of $W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

- Players U and U' both use ek and agreed on pub.
- U, with $(\mathsf{priv}, priv', W)$, generates $\mathsf{hk} = \mathsf{HashKG}(\mathsf{ek}, \mathsf{pub})$ and $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathsf{ek}, \mathsf{pub}), \perp)$.
  U computes $\ell = (U, U', \mathsf{hp})$ and $C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r)$, with r a random scalar in $\mathbb{Z}_p$, and sends hp, C to U'.
- Upon receiving hp', C' from U', it sets $\ell' = (U', U, \mathsf{hp}')$,
  U computes $H = \mathsf{Hash}(\mathsf{hk}, ((\mathsf{ek}, \mathsf{pub}), (priv', \mathsf{priv})), (\ell', C'))$,
  $H' = \mathsf{ProjHash}(\mathsf{hp}', ((\mathsf{ek}, \mathsf{pub}), (\mathsf{priv}, priv')), (\ell, C), r)$, and $\mathsf{sk} = H \cdot H'$.

For $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$ and $\mathsf{aux} = (\mathsf{priv}, priv')$,

$$\textsc{Lof}\mathrm{C}_{\mathsf{crs},\mathsf{aux}} = \left\{ (\ell, C) \;\middle|\; \begin{array}{ll} \exists r, \exists W, & C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r) \\ & \text{and } W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}} \end{array} \right\}.$$

Figure I.2: One-Round LAKE

We stress that hk and hp can depend on crs but not on aux, hence the notations used in the Figure I.2. Using a similar proof as in [KV11], one can state the following theorem (more details on the security model and the full proof can be found in I.B):

Theorem I.5.2 If the encryption scheme is IND-CCA, and $\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')}$ languages admit KV-SPHFs, then our LAKE protocol is secure.

From LAKE to PAKE.

One can remark that this theorem immediately proves the security of our PAKE from Figure I.1: one uses $\mathsf{priv} = \mathsf{priv}' = \mathsf{pw}$ and $\mathsf{pub} = \emptyset$, for the language of the ciphertexts of pw.

I.5.2 Two-Flow Waters Blind Signature

Blind signature schemes, introduced by Chaum in 1982 [Cha82], allow a person to get a signature by another party without revealing any information about the message being signed. A blind signature can then be publicly verified using the unblinded message.

In [BPV12b], the authors presented a technique to do efficient blind signatures using an SPHF: it is still the most efficient Waters blind signature known so far. In addition, the resulting signature is a classical Waters signature (see I.C.1 for the definition of Waters signatures).

The construction basically consists in encrypting the message bit-by-bit under distinct bases, that will allow the generation of a masked Waters hash of the message. Thereafter, the signer will easily derive a masked signature the user will eventually unmask. However, in order to generate the masked signature, the signer wants some guarantees on the ciphertexts, namely that some ciphertexts contain a bit (in order to allow extractability) and that another ciphertext contains a Diffie-Hellman value. Using our new techniques, we essentially improve on the proof of bit encryption by using the above randomness-reuse technique.

Definition.

Before showing our new construction, let us first recall the definition of blind signatures.

A blind signature scheme $\mathcal{BS}$ is defined by three algorithms $(\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSVerif})$ and one interactive protocol $\mathsf{BSProtocol}\langle \mathcal{S}, \mathcal{U} \rangle$:

- $\mathsf{BSSetup}(1^{\mathfrak{K}})$, generates the global parameters param of the system;

- $\mathsf{BSKeyGen}(\mathsf{param})$ is a probabilistic polynomial-time algorithm that generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$ where vk is the public (verifying) key and sk is the secret (signing) key;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$: this is a probabilistic polynomial-time interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, M)$, for a message $M \in \{0,1\}^n$. It generates a signature σ on M under vk related to sk for the user.
- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$ is a deterministic polynomial-time algorithm which outputs 1 if the signature σ is valid with respect to m and vk, 0 otherwise.

A blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: blindness and unforgeability.

Blindness states that a malicious signer should be unable to decide which of two messages m_0, m_1 has been signed first in two *valid* executions with an honest user.
Note that the malicious signer $\mathcal{A}$ can choose arbitrarily the keys and thus the verification key vk given to users. However, if $\mathcal{A}$ refuses to sign one of the inputs (*i.e.* $\sigma_i = \perp$ for $i \in \{0,1\}$) or if one of the signatures is invalid (i.e. $\mathsf{BSVerif}(\mathsf{vk}, m_i, \sigma_i) = 0$ for $i \in \{0,1\}$) then the two resulting signatures are set to $\perp$; the adversary therefore does not gain any advantage if he decides to prevent the normal game execution.

$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk}, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $\sigma_b \leftarrow \mathsf{BSProtocol}\langle\mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$
4. $\sigma_{1-b} \leftarrow \mathsf{BSProtocol}\langle\mathcal{A}, \mathcal{U}(\mathsf{vk}, m_{1-b})\rangle$
5. $b^* \leftarrow \mathcal{S}^*(\texttt{GUESS} : \sigma_0, \sigma_1)$;
6. `RETURN` $b^* = b$.

The advantages are

$$\begin{aligned}\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) &= \Pr[\mathsf{Exp}^{\mathsf{bl}-1}_{\mathcal{BS},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{bl}-0}_{\mathcal{BS},\mathcal{A}}(k) = 1]\\ \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS}}(k,t) &= \max_{\mathcal{A}\le t} \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E},\mathcal{A}}(k).\end{aligned}$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{A}}(k)$ for $b \in \{0,1\}$ runs in time at most t. The scheme $\mathcal{BS}$ is deemed blind, if for all polynomials p, $\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{E}}(k, p(k))$ is a negligible function of k.

In the security game, we insist on *valid* executions which end with a valid signature σ of the message used by $\mathcal{U}$ under the key vk. The signer could of course send a wrong answer which would lead to an invalid signature. Then, it could easily distinguish a valid signature from an invalid one, and thus the two executions. But this is a kind of denial of service, that is out of scope of this work. This thus means that one valid execution is indistinguishable from other valid executions. This notion was formalized in [HKKL07] and termed *a posteriori blindness*. We enforce this requirement and we add the constraint that even if the signer may deviate arbitrarily from the $\mathsf{BSProtocol}\langle\mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$ protocol specification (for $b \in \{0,1\}$) in an attempt to cheat, the signatures σ_0 and σ_1 must be valid with overwhelming probability (i.e. $\mathsf{BSVerif}(\mathsf{vk}, m_0, \sigma_0) = \mathsf{BSVerif}(\mathsf{vk}, m_1, \sigma_1) = 1$ except with negligible probability).

In out model, adversaries are willing to actively cheat but only if they are not caught. It is relevant in contexts where honest behavior cannot be assumed, but where the companies, institutions and individuals involved cannot afford the embarrassment, loss of reputation, and negative press associated with being caught cheating (e.g. e-cash or e-voting). This is similar to the notion of security against covert adversaries from [AL10] and we call this notion *blindness against covert adversaries*.

An adversary against the (one-more) unforgeability tries to generate $q+1$ valid signatures after at most q complete interactions with the honest signer. This security notion can be formalized by the security game $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{U}^*}(k)$ where the adversary is permitted to keep some internal state between the various calls $\texttt{INIT}_i$ (for $i \in \{1,\ldots,q_s\}$), `FIND` and `GUESS`.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k)$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^k)$
2. $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. For $i = 1,\ldots,q_s$, $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{A}(\texttt{INIT}_i : \mathsf{vk})\rangle$
4. $((m_1,\sigma_1),\ldots,(m_{q_s+1},\sigma_{q_s+1})) \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{vk})$;
5. `IF` $\exists i \neq j, m_i = m_j$ `OR` $\exists i, \mathsf{Verif}(\mathsf{pk}, m_i, \sigma_i) = 0$ `RETURN` 0
6. `ELSE RETURN` 1

The success probabilities are

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{BS},\mathcal{A}}(k) = 1] \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$$

where the maximum is over all $\mathcal{A}$ such that the random experiments $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(k)$ runs in time at most t. The scheme $\mathcal{S}$ is deemed $\mathsf{EUF}-\mathsf{CMA}$-secure, if for all polynomial p, $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,p(k))$ is a negligible function of k.

Concurrency in the context of blind signatures was put forth by Juels, Luby and Ostrovsky [JLO97] who presented the first security model for blind signatures that takes into account that the adversary may launch many concurrent sessions of the blind signing protocol (operating as either the user or the signer). In this paper, we consider only *round-optimal* blind signatures (i.e. the user sends a single message to the signer and gets a single response) which are concurrently secure.

Construction.

Here, we give a sketch of the protocol (in which i always ranges from 1 to ℓ, except if stated otherwise) and its communication cost:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates a pairing-friendly system $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e;g_1,g_2)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, a random generator $h_s \in \mathbb{G}_1$ as well as independent generators $\vec{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters hash function $\mathcal{F}(M) = u_0 \prod_i u_i^{M_i}$, for $M = (M_i)_i \in \{0,1\}^\ell$, and finally random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$. It also sets $\mathsf{ek} = (h_i)_i = (g_1^{x_i})_i$ and $g_s = \prod_i h_i$. It outputs the global parameters $\mathsf{param} = (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,g_1,g_2,\mathsf{ek},g_s,h_s,\vec{u})$. Essentially, g_1 and ek compose the encryption key for an ElGamal ciphertext on a vector, applying the randomness-reuse technique, while g_s, g_2 and h_s are the bases used for the Waters signature;
- $\mathsf{KeyGen}(\mathsf{param})$ picks at random $x \in \mathbb{Z}_p$, sets the signing key $\mathsf{sk} = h_s^x$ and the verification key $\mathsf{vk} = (g_s^x, g_2^x)$;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{U}(\mathsf{vk},M)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on $M = (M_i)_i \in \{0,1\}^\ell$:
  - Message Encryption: $\mathcal{U}$ chooses a random $r \in \mathbb{Z}_p$ and encrypts $u_i^{M_i}$ for all the i's with the same random r: $c_0 = g_1^r$ and $(c_i = h_i^r u_i^{M_i})_i$. $\mathcal{U}$ also encrypts vk_1^r, into $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$, with a different random s: It eventually sends $(c_0,(c_i)_i,(d_0,d_1)) \in \mathbb{G}_1^{\ell+3}$;

– Signature Generation: $\mathcal{S}$ first computes the masked Waters hash of the message $c = u_0 \prod_i c_i = (\prod_i h_i)^r \mathcal{F}(M) = g_s^r \mathcal{F}(M)$, and generates the masked signature $(\sigma'_1 = h_s^x c^t = h_s^x g_s^{rt} \mathcal{F}(M)^t, \sigma_2 = (g_s^t, g_2^t))$ for a random $t \xleftarrow{\$} \mathbb{Z}_p$;

– SPHF: $\mathcal{S}$ needs the guarantee that each ElGamal ciphertext (c_0, c_i) encrypts either 1 or u_i under the key (g_1, h_i), and (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$ under the key (g_1, h_1).
The signer chooses a random $\mathsf{hk} = (\eta, (\theta_i)_i, (\nu_i)_i, \gamma, (\mu_i)_i, \lambda)$ and sets $\mathsf{hp}_1 = g_1^\eta \cdot \prod_i h_i^{\theta_i} \cdot \mathsf{vk}_1^\lambda$, $(\mathsf{hp}_{2,i} = u_i^{\theta_i} c_0^{\nu_i} (c_i/u_i)^{\mu_i})_i$, $(\mathsf{hp}_{3,i} = g_1^{\theta_i} h_i^{\mu_i})_i$, and $\mathsf{hp}_4 = g_1^\gamma h_1^\lambda$, then $H = c_0^\eta \cdot \prod_i c_i^{\theta_i} \cdot d_0^\gamma \cdot d_1^\lambda = \mathsf{hp}_1^r \cdot \prod_i \mathsf{hp}_{2,i}^{M_i} \cdot \mathsf{hp}_{3,i}^{-rM_i} \cdot \mathsf{hp}_4^s = H' \in \mathbb{G}_1$. This SPHF is easily obtained from the above GL-SPHF on bit encryption, as shown in I.C;

– Masked Signature: $\mathcal{S}$ sends $(\mathsf{hp}, \Sigma = \sigma'_1 \cdot H, \sigma_2) \in \mathbb{G}_1^{2\ell+3} \times \mathbb{G}_2$;

– Signature Recovery: Upon receiving $(\mathsf{hp}, \Sigma, \sigma_2)$, using his witnesses and hp, $\mathcal{U}$ computes H' and unmasks σ'_1. Thanks to the knowledge of r, it can compute $\sigma_1 = \sigma'_1 \cdot (\sigma_{2,1})^{-r}$. Note that if $H' = H$, then $\sigma_1 = h_s^x \mathcal{F}(M)^t$, which together with $\sigma_2 = (g_s^t, g_2^t)$ is a valid Waters signature on M;

- $\mathsf{Verif}(\mathsf{vk}, M, (\sigma_1, (\sigma_{2,1}, \sigma_{2,2}))$, checks whether both $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$ and $e(\sigma_1, g_2) = e(h, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$ are satisfied or not.

Security Proof.

The security proof is similar to the one in [BPV12b] and is given in I.C.2.

Complexity.

The whole process requires only $3\ell + 7$ elements in $\mathbb{G}_1$ ($\ell + 3$ for the ciphertexts, $2\ell + 4$ for the projection key, Σ and $\sigma_{2,1}$) and 1 in $\mathbb{G}_2$ ($\sigma_{2,2}$). This is more efficient than the instantiation from [BPV12b] ($5\ell + 6$ elements in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$) already using an SPHF, and much more efficient than the instantiation from [BFPV11] ($6\ell + 7$ elements in $\mathbb{G}_1$ and $6\ell + 5$ in $\mathbb{G}_2$) using a Groth-Sahai [GS08] NIZK proof.

I.A Preliminaries

I.A.1 Formal Definitions of the Basic Primitives

We first recall the definitions of some of the basic tools, with the corresponding security notions and their respective success/advantage.

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions $\mathfrak{H}_K$ from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^\mathfrak{K}$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\begin{aligned} \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) &= \Pr[\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(\mathfrak{H}_K) : \mathfrak{H}_K(m_0) = \mathfrak{H}_K(m_1)], \\ \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) &= \max_{\mathcal{A} \le t} \{\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A})\}, \end{aligned}$$

where the latter notation means the maximum over the adversaries running within time t.

Labeled Encryption Scheme. A labeled public-key encryption scheme $\mathcal{E}$ is defined by four algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key ek and the decryption key dk;
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$ for an invalid ciphertext.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)) = m.$$

- *Indistinguishability under chosen-ciphertext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls `FIND` and `GUESS`, and makes use of the oracle $\mathsf{ODecrypt}$:
  - $\mathsf{ODecrypt}(\ell, c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key dk. The input queries (ℓ, c) are added to the list $\mathcal{CT}$.

  The advantages are

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1] \\ \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(t) &= \max_{\mathcal{A}\leq t}\{\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A})\}. \end{aligned}$$

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\ell^*, \mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot,\cdot))$
6. `IF` $(\ell^*, c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

I.A.2 Statistical and Computational Distances

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two probability distributions over a finite set $\mathcal{S}$ and let X and Y be two random variables with these two respective distributions.

Statistical Distance.

The statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is also the statistical distance between X and Y:

$$\mathrm{Dist}(\mathcal{D}_1, \mathcal{D}_2) = \mathrm{Dist}(X, Y) = \sum_{x \in \mathcal{S}} \left| \Pr[\, X = x \,] - \Pr[\, Y = x \,] \right|.$$

If the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is less than or equal to ε, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_1$ and D_2 are 0-close, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are perfectly indistinguishable.

Computational Distance.

We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t,ε)-computationally indistinguishable, if, for every probabilistic algorithm $\mathcal{A}$ running in time at most t:

$$|\Pr[\mathcal{A}(X)=1] - \Pr[\mathcal{A}(Y)=1]| \leq \varepsilon.$$

We can note that for any t and ε, $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t,ε)-computationally indistinguishable, if they are ε-close.

I.A.3 Concrete Instantiations

All the analyses in this paper could be instantiated with ElGamal-like schemes, based on either the Decisional Diffie-Hellman (DDH) assumption, or the Decisional Linear (DLin) assumption. But we focus on the former only:

Definition I.A.1 [Decisional Diffie-Hellman (DDH)] The Decisional Diffie-Hellman assumption says that, in a group $(p,\mathbb{G},g)$, when we are given (g^a,g^b,g^c) for unknown random $a,b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{ddh}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple within time t.

Cramer-Shoup (CS) Encryption Scheme [CS98]:

it can be turned into a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1,g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1,x_2,y_1,y_2,z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1}g_2^{x_2}$, $d = g_1^{y_1}g_2^{y_2}$, and $h = g_1^z$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1,g_2,c,d,h,\mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell,\mathsf{ek},M;r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^{\xi})^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell,\mathbf{u},e)$.
- $\mathsf{Decrypt}(\ell,\mathsf{dk},C)$: one first computes $\xi = \mathfrak{H}_K(\ell,\mathbf{u},e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/u_1^z$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and the collision-resistance / universal one-wayness of the hash function $\mathcal{H}$.

I.B Security Proof for LAKE

I.B.1 Security Model

In this paper, we focus on efficiency and propose (in Section I.5.1) an extension of the PAKE security model presented by Bellare-Pointcheval-Rogaway [BPR00] model for PAKE, between n players in the presence of an adversary. The adversary $\mathcal{A}$ plays a find-then-guess game against n players $(P_i)_{i=1,\ldots,n}$. It has access to several instances Π^s_U for each player $U \in \{P_i\}$ and can activate them (in order to model concurrent executions) via several queries, described below:

- $\mathsf{Execute}(U, s, U', t)$: one outputs the transcript of an execution of the protocol between the instance Π_U^s of U and the instance $\Pi_{U'}^t$ of U'. It models passive eavesdropping attacks;
- $\mathsf{Send}(U, s, U', t, m)$: one sends the message m to the instance $\Pi_{U'}^t$ of U' in the name of the instance Π_U^s of U. It models active attacks;
- $\mathsf{Reveal}(U, s)$: if the instance Π_U^s of U has "accepted", one outputs the session key, otherwise one outputs $\perp$. It models a possible bad later use of the session key;
- $\mathsf{Test}(U, s)$: one first flips a coin $b \xleftarrow{\$} \{0, 1\}$, if $b = 1$ one outputs $\mathsf{Reveal}(U, s)$, otherwise one outputs a truly random key. It models the secrecy of the session key.

We say that Π_U^s and $\Pi_{U'}^t$ have matching conversations if inputs-outputs of the former correspond to the outputs-inputs of the latter and vice-versa. They are then called *partners*. We say that an instance is *fresh* if the key exists and is not trivially known by the adversary: more precisely, Π_U^s is fresh if

- Π_U^s has *accepted* the session, which is required to compute a session key;
- Π_U^s has not been asked a Reveal-query;
- no $\Pi_{U'}^t$ with *matching conversations* with Π_U^s has been asked a Reveal-query.

A key exchange protocol is then said secure if keys are indistinguishable from random keys for adversaries. Formally, the adversary is allowed to ask as many $\mathsf{Execute}$, Send and Reveal-queries as it likes, and then only one Test-query to a *fresh* instance Π_U^s of a player. The adversary wins if it has guessed correctly the bit b in this query.

I.B.2 Proof of Theorem I.5.2

This proof follows the one from [KV11]. It starts from the real attack game, in a Game 0: $\mathsf{Adv}_0(\mathcal{A}) = \varepsilon$. We incrementally modify the simulation to make possible the trivial attacks only. In the first games, all the honest players have their own values, and the simulator knows and can use them. Following [KV11], we can assume that there are two kinds of Send-queries: $\mathsf{Send}_0(U, s, U')$-queries where the adversary asks the instance Π_U^s to initiate an execution with an instance of U'. It is answered by the flow U' should send to communicate with U; $\mathsf{Send}_1(U, s, m)$-queries where the adversary sends the message m to the instance Π_U^s. It gives no answer back, but defines the session key, for possible later Reveal or Test-queries.

Game G_0: We first modify the way $\mathsf{Execute}$-queries are answered: we replace C and C' by encryptions of a fixed message M_0, that parses as two private parts P and P' and a word W, such that W is not in the language induced by (pub, P). Since the hashing keys are known, the common session key is computed as

$$\mathsf{sk} = \mathsf{Hash}(\mathsf{hk}, ((\mathsf{ek}, \mathsf{pub}), priv'), C') \times \mathsf{Hash}(\mathsf{hk}', ((\mathsf{ek}, \mathsf{pub}), \mathsf{priv}), C).$$

Since we could have first modified the way to compute sk, that has no impact at all from the soundness of the SPHF, the unique difference comes from the different ciphertexts. This is anyway indistinguishable under the $\mathsf{IND\text{-}CPA}$ property of the encryption scheme, for each $\mathsf{Execute}$-query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_0(\mathcal{A}) - \mathsf{Adv}_{-1}(\mathcal{A})| \leq \mathsf{negl}()$.

Game G_1: We modify again the way $\mathsf{Execute}$-queries are answered: we replace the common session key by a truly random value. Since the languages are not satisfied, the smoothness

guarantees indistinguishability: $|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_0(\mathcal{A})| \leq \mathsf{negl}()$.

Game G_2: We now modify the way one answers the Send_1-queries, by using a decryption oracle, or alternatively knowing the decryption key. More precisely, when a message (hp, C) is sent, three cases can appear:

- it has been generated (altered) by the adversary, then one first decrypts the ciphertext to get $(\mathsf{priv}', priv, W')$ used by the adversary. Then
  - If they are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = priv', priv = \mathsf{priv}$) —event Ev— one declares that $\mathcal{A}$ succeeds (saying that $b' = b$) and terminates the game;
  - if they are not both correct and consistent with the receiver's values, one chooses sk at random.
- it is a replay of a previous flow sent by the simulator, then, in particular, one knows the hashing keys, and one can compute the session keys using all the hashing keys.

The first case can only increase the advantage of the adversary in case Ev happens (which probability is computed in G_5). The second change is indistinguishable under the adaptive-smoothness and thus only increases the advantage of the adversary by a negligible term. The third change does not affect the way the key is computed, so finally: $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{negl}()$.

Game G_3: We modify again the way one answers the Send_1-queries. More precisely, when a message (hp, C) is sent, two cases can appear:

- if there is an instance $\Pi^t_{U'}$ partnered with Π^s_U that receives this flow, then set the key identical to the key for $\Pi^t_{U'}$;
- otherwise, one chooses sk at random.

The former case remains identical since the message is a replay of a previous flow, and the latter is indistinguishable, as in [KV11], thanks to the adaptive-smoothness and their technical lemma that proves that all the hash values are random looking even when hashing keys and ciphertexts are re-used: $|\mathsf{Adv}_3(\mathcal{A}) - \mathsf{Adv}_2(\mathcal{A})| \leq \mathsf{negl}()$.

Game G_4: We now modify the way one answers the Send_0-queries: instead of encrypting the correct values, one does as in G_0 for $\mathsf{Execute}$-queries, and encrypts M_0. Since for simulating the Send_1-queries decryptions are required, indistinguishability relies on the $\mathsf{IND\text{-}CCA}$ security of the encryption scheme: $|\mathsf{Adv}_4(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq \mathsf{negl}()$.

Game G_5: For all the hashing and projection keys, we now use the dummy private inputs. Since we restricted hk and hp not to depend on aux, the distributions of these keys are independent of the auxiliary private inputs: $|\mathsf{Adv}_5(\mathcal{A}) - \mathsf{Adv}_4(\mathcal{A})| \leq \mathsf{negl}()$.

If one combines all the relations, one gets $\mathsf{Adv}_5(\mathcal{A}) \geq \mathsf{Adv}_0(\mathcal{A}) - \mathsf{negl}() = \varepsilon - \mathsf{negl}()$.

One can note that in this final game, the values of the honest players are not used anymore during the simulation, but just for declaring whether the adversary has won or not (event Ev). Otherwise, non-partnered players have random and independent keys, and thus unless the simulator stops the simulation, the advantage in the last game is exactly 0: $\mathsf{Adv}_5(\mathcal{A}) = \Pr[\mathsf{Ev}]$. And thus, we have $\varepsilon \leq \Pr[\mathsf{Ev}] + \mathsf{negl}()$.

Let us recall that Ev means that the adversary has encrypted $(\mathsf{priv}', priv, W')$ that are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = priv', priv = \mathsf{priv}$). Since the values for the honest players are never used during the simulation, we can assume we choose them at the very end only to check whether event Ev happened:

$$\Pr[\mathsf{Ev}] = \Pr[\exists k : \mathsf{priv}'(k) = priv'_{i_k}, priv(k) = priv_{i_k}, W'(k) \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'_{i_k}}]$$

where k lists all the Send_1-queries with adversary-generated messages in which the ciphertexts decrypt to $(\mathsf{priv}'(k), priv(k), W'(k))$, and i_k is the index of the recipient of k-th Send_1-query: it has first to guess the private values, and then once it has guessed them it has to find a word in the language:

$$\Pr[\mathsf{Ev}] \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t),$$

where m is the minimal min-entropy on the joint distributions of the $(\mathsf{priv}, priv')$ for any two players U, U' who want to communicate, and $\mathsf{Succ}^{\mathrm{L}}(t)$ is the best success an adversary can get in finding a word in a language $\mathrm{L}_{\mathsf{pub},\mathsf{priv}}$. Then, by combining all the inequalities, one gets

$$\varepsilon \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t) + \mathsf{negl}().$$

I.C Blind Signature

In this appendix, we give details on our two-flow Waters blind signature scheme outlined in Section I.5.2. We first present the asymmetric variant of Waters signatures proposed in [BFPV11] and then recall the formal security definitions of blind signatures and of their security properties. Using the formalism from I.D, we describe in details the SPHF used in the scheme and finally prove the security of our scheme.

I.C.1 Waters Signature (Asymmetric Setting)

In 2011, Blazy, Fuchsbauer, Pointcheval and Vergnaud [BFPV11] proposed the following variant of Waters signatures in an asymmetric pairing-friendly environment:

- $\mathsf{Setup}(1^{\mathfrak{K}})$: in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$, one chooses a random vector $\vec{u} = (u_0, \ldots, u_\ell) \xleftarrow{\$} \mathbb{G}_1^{\ell+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{\ell} u_i^{M_i}$ for $M = (M_i)_i \in \{0,1\}^\ell$. We also need two extra generators $(g_s, h_s) \xleftarrow{\$} \mathbb{G}_1^2$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, g_s, h_s, \vec{u})$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (g_s^x, g_2^x) = (\mathsf{vk}_1, \mathsf{vk}_2)$, and the secret key is set as $\mathsf{sk} = h_s^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $t \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = (\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^t, \sigma_2 = (\sigma_{2,1} = g_s^t, \sigma_{2,2} = g_2^t))$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g_2) = e(h_s, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$, and $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$:

Definition I.C.1 [The Advanced Computational Diffie-Hellman problem (CDH^+)] In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

I.C.2 Underlying SPHF in the Blind Signature Scheme

Following [BPV12b], our scheme makes use of an SPHF in the interactive signing protocol to insure (in an efficient way) that the user actually knows the signed message. As outlined in Section I.5.2, during the interactive process of the blind signature protocol, we have:

- General setting: a pairing-friendly system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;
- Encryption parameters: random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$ with $(h_i = g_1^{x_i})_i$, where i ranges from 1 to ℓ, as everywhere in the following. Then, $\mathsf{ek} = (h_i)_i$;
- Signature parameters: independent generators $\vec{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters function, $g_s = \prod_i h_i$, and a random generator $h_s \in \mathbb{G}_1$, then $\mathsf{sk} = h_s^x$ and $\mathsf{vk} = (g_s^x, g_2^x)$, for a random scalar x.

The user has generated $c_0 = g_1^r$ and $c_i = h_i^r u_i^{M_i}$ for $i = 1, \ldots, \ell$, as well as $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$. In the following simulation, we will extract $(M_i)_i$ from $C = (c_0, (c_i)_i)$, and we thus need to be sure that this message can be extracted. In addition, the simulator will also need to know vk_1^r to generate the blinded signature, hence its encryption in (d_0, d_1). But this has to be checked, with the following language membership, where we use notations from I.D:

1. each (c_0, c_i) encrypts a bit;

$$\Gamma(C) = \left(\begin{array}{c|ccc|ccc|ccc} g_1 & h_1 & \ldots & h_\ell & 1 & \ldots & 1 & 1 & \ldots & 1 \\ \hline 1 & u_1 & & 1 & c_0 & & 1 & c_1/u_1 & & 1 \\ \vdots & & \ddots & & & \ddots & & & \ddots & \\ 1 & 1 & & u_\ell & 1 & & c_0 & 1 & & c_\ell/u_\ell \\ \hline 1 & & 1 & & g_1 & & 1 & h_1 & & 1 \\ \vdots & & & & & \ddots & & & \ddots & \\ 1 & & & & 1 & & g_1 & 1 & & h_\ell \end{array}\right)$$

and

$$\begin{array}{rcl} \Theta_{\mathsf{aux}}(C) & = & (c_0, (c_i)_i, (1)_i, (1)_i) \\ \vec{\lambda} & = & (r, (M_i)_i, (-rM_i)_i) \\ \vec{\lambda} \cdot \Gamma(C) & = & (g_1^r, (h_i^r u_i^{M_i})_i, (c_0^{M_i} g_1^{-rM_i})_i, ((c_i/u_i h_i^r)^{M_i})_i). \end{array}$$

2. the ciphertext (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$;

$$\Gamma = \begin{pmatrix} g_1 & 1 & \mathsf{vk}_1 \\ 1 & g_1 & h_1 \end{pmatrix} \quad \begin{array}{rcl} \Theta_{\mathsf{aux}}(C) & = & (c_0, d_0, d_1) \\ \vec{\lambda} \cdot \Gamma & = & (g_1^r, g_1^s, \mathsf{vk}_1^r h_1^s) \end{array} \quad \vec{\lambda} = (r, s)$$

The two matrices can be compressed with a common row/column: the same witness r is indeed used in both matrices, the two corresponding rows can be merged; the first column is the same in both matrices, it can thus be a common one:

$$\Gamma(C) = \left(\begin{array}{c|ccc|ccc|c|ccc|c} g_1 & h_1 & \ldots & h_\ell & 1 & \ldots & 1 & 1 & 1 & \ldots & 1 & \mathsf{vk}_1 \\ \hline 1 & u_1 & & 1 & c_0 & & 1 & 1 & c_1/u_1 & & 1 & 1 \\ \vdots & & \ddots & & & \ddots & & \vdots & & \ddots & & \vdots \\ 1 & 1 & & u_\ell & 1 & & c_0 & 1 & 1 & & c_\ell/u_\ell & 1 \\ \hline 1 & & 1 & & g_1 & & 1 & 1 & h_1 & & 1 & 1 \\ \vdots & & & & & \ddots & & \vdots & & \ddots & & \vdots \\ 1 & & & & 1 & & g_1 & 1 & 1 & & h_\ell & 1 \\ \hline 1 & 1 & \ldots & 1 & 1 & \ldots & 1 & g_1 & 1 & \ldots & 1 & h_1 \end{array}\right)$$

and

$$\begin{aligned}\Theta_{\mathsf{aux}}(C) &= (c_0,(c_i)_i,(1)_i,d_0,(1)_i,d_1)\\ \vec{\lambda} &= (r,(M_i)_i,(-rM_i)_i,s)\\ \vec{\lambda}\cdot\Gamma(C) &= (g_1^r,(h_i^r u_i^{M_i})_i,(c_0^{M_i}g_1^{-rM_i})_i,g_1^s,((c_i/u_ih_i^r)^{M_i})_i,\mathsf{vk}_1^r h_1^s).\end{aligned}$$

This leads to, with $\mathsf{hk}=(\eta,\{\theta_i\}_i,\{\nu_i\}_i,\gamma,\{\mu_i\}_i,\lambda)$,

$$\mathsf{hp}_1 = g_1^\eta\cdot\prod\nolimits_i h_i^{\theta_i}\cdot\mathsf{vk}_1^\lambda \qquad (\mathsf{hp}_{2,i}=u_i^{\theta_i}c_0^{\nu_i}(c_i/u_i)^{\mu_i})_i \qquad (\mathsf{hp}_{3,i}=g_1^{\nu_i}h_i^{\mu_i})_i \qquad \mathsf{hp}_4=g_1^\gamma h_1^\lambda$$

$$H=c_0^\eta\cdot\prod\nolimits_i c_i^{\theta_i}\cdot d_0^\gamma\cdot d_1^\lambda=\mathsf{hp}_1^r\cdot\prod\nolimits_i\mathsf{hp}_{2,i}^{M_i}\cdot\mathsf{hp}_{3,i}^{-rM_i}\cdot\mathsf{hp}_4^s=H'.$$

The signers thus uses H to mask his blinded signature (σ_1',σ_2). But since σ_2 is just a random pair, only σ_1' needs to be masked. Without it, one cannot forge a signature, but it can be unmasked by the user with H', if the values $(c_0,(c_i)_i,(d_0,d_1))$ are in the correct language, and thus are correct ciphertexts.

One can note that the projection key consists of $2\ell+2$ group elements in $\mathbb{G}_1$, and the hash value is in $\mathbb{G}_1$. No pairings are needed for this SPHF. Since Γ depends on C, this is a GL-SPHF, but this is enough for our interactive protocol.

I.C.3 Security proofs

Proposition I.C.2 Our blind signature scheme is blind against covert adversaries under the DDH assumption in $\mathbb{G}_1$[2]:

$$\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K})\le 2\times(\ell+1)\times\mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}).$$

Proof: Let us consider an adversary $\mathcal{A}$ against the blindness of our scheme. We build an adversary $\mathcal{B}$ against the DDH assumption in $\mathbb{G}_1$.

Game G_0: In a first game G_0, we run the standard protocol:

- $\mathsf{BSSetup}(1^k)$, $\mathcal{B}$ generates $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e)$ with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. It also generates independent generators $\vec{u}=(u_i)_{i\in\{0,\dots,\ell\}}\in\mathbb{G}_1^{\ell+1}$ for the Waters function and sets $\mathsf{ek}=(h_i)_i$ and $g_s=\prod_i h_i$. It generates $h_s=g_s^\alpha\in\mathbb{G}_1$ and defines the global parameters as $\mathsf{param}=(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,g_1,g_2,\mathsf{ek},g_s,h_s,\vec{u})$;
- The adversary $\mathcal{A}$ generates a verification key $\mathsf{vk}=(\mathsf{vk}_1,\mathsf{vk}_2)\in\mathbb{G}_1\times\mathbb{G}_2$ such that $e(\mathsf{vk}_1,g_2)=e(g_s,\mathsf{vk}_2)$ and two ℓ-bit messages M^0,M^1.
- $\mathcal{A}$ and $\mathcal{B}$ run twice the interactive issuing protocol, first on the message M^b, and then on the message M^{1-b}:
  - $\mathcal{B}$ chooses a random $r\in\mathbb{Z}_p$ and encrypts $u_i^{M_i}$ for all the i's with the same random r: $c_0=g_1^r$ and $(c_i=h_i^r u_i^{M_i^b})_i$. $\mathcal{B}$ also encrypts vk_1^r, into $d_0=g_1^s$, $d_1=h_1^s\mathsf{vk}_1^r$ and sends $(c_0,(c_i)_i,(d_0,d_1))$ to $\mathcal{A}$.
  - $\mathcal{A}$ then outputs $(\mathsf{hp},\Sigma=\sigma_1'\times H,\sigma_2)$
  - $\mathcal{B}$, using its witnesses and hp, computes H' and unmasks $\sigma_1'=\Sigma/H$ which together with σ_2 should be a valid Waters Signature on M^b. It then randomizes the signature with s' to get Σ_b.

  The same is done a second time with M^{1-b} to get Σ_{1-b}.

[2]This assumption is sometimes referred to as the XDH assumption.

- $\mathcal{B}$ publishes (Σ_0, Σ_1).
- Eventually, $\mathcal{A}$ outputs b'.

We denote by ε the advantage of $\mathcal{A}$ in this game. By definition, we have:

$$\varepsilon = \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) \quad = \quad \Pr_{\boldsymbol{G}_0}[b' = 1 | b = 1] - \Pr_{\boldsymbol{G}_0}[b' = 1 | b = 0] = 2 \times \Pr_{\boldsymbol{G}_0}[b' = b] - 1.$$

Game $\boldsymbol{G}_1$: In a second game $\boldsymbol{G}_1$, we modify the way $\mathcal{B}$ extracts the signatures Σ_b and Σ_{1-b}. Since $\mathcal{B}$ knows the scalar α such that $h_s = g_s^\alpha$ it can compute the secret key $\mathsf{sk} = \mathsf{vk}_1^\alpha$ associated to $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2)$. One can note that, since we focus on valid executions with the signer, and due to the re-randomization of Waters signatures which leads to random signatures, $\mathcal{B}$ can generates itself random signatures on M^b and M^{1-b} using sk. This game is perfectly indistinguishable from the previous one:

$$\Pr_{\boldsymbol{G}_1}[b' = b] = \Pr_{\boldsymbol{G}_0}[b' = b].$$

Game $\boldsymbol{G}_2$: In this final game, we replace all the ciphertexts sent by $\mathcal{B}$ by encryption of random group elements in $\mathbb{G}_1$. For proving indistinguishability with the previous game, we use the hybrid technique for ElGamal ciphertexts with randomness re-use [BBS03]:

$$\varepsilon \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}) + 2 \times \Pr_{\boldsymbol{G}_2}[b' = b] - 1.$$

In this last game, the two executions are thus perfectly indistinguishable, and thus $\Pr_{\boldsymbol{G}_2}[b' = b] = 1/2$ and we get the bound claimed in the proposition. ∎

Proposition I.C.3 Our blind signature scheme is unforgeable under the CDH^+ assumption.

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K}) \leq \Theta\left(\frac{\mathsf{Succ}^{\mathsf{CDH}^+}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})}{q_s\sqrt{\ell}}\right).$$

Proof: Let $\mathcal{A}$ be an adversary against the Unforgeability of the scheme. We assume that this adversary is able after q_s signing queries to output at least $q_s + 1$ valid signatures on different messages (for some q_s polynomial in the security parameter). We now build an adversary $\mathcal{B}$ against the CDH^+ assumption.

- $\mathcal{B}$ is first given a CDH^+ challenge $(g_s, g_2, g_s^x, g_2^x, h_s)$ in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$
- $\mathcal{B}$ emulates $\mathsf{BSSetup}$: it picks a random position $j \xleftarrow{\$} \{0, \ldots, \ell\}$, random indices $y_0, \ldots, y_\ell \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$ and random scalars $z_0, \ldots, z_\ell \xleftarrow{\$} \mathbb{Z}_p$ and publishes $\vec{u} = (u_i)_{i \in \{0,\ldots,\ell\}} \in \mathbb{G}^{\ell+1}$ for the Waters function, where $u_0 = h_s^{y_0 - 2jq_s} g_s^{z_0}$ and $u_i = h_s^{y_i} g_s^{z_i}$ for $i \in \{1, \ldots, \ell\}$. It sets $g_1 = g_s^\gamma$ and $\mathsf{ek} = (h_i)_i$ with $h_i = g_1^{a_i} \in \mathbb{G}_1$ for $i \in \{1, \ldots, \ell\}$ for some known random scalars $a_1, \ldots, a_\ell$ and $\gamma = 1/\sum_i a_i \bmod p$. It keeps secret the associated decryption key $\mathsf{dk} = (a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell$ and outputs the global $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \vec{u})$.
- $\mathcal{B}$ then emulates $\mathsf{BSKeyGen}$: it publishes $\mathsf{vk} = (g_s^x, g_2^x)$ from the challenge as its verification key (one can note that recovering the signing key h_s^x is the goal of our adversary $\mathcal{B}$);

- $\mathcal{A}$ can now interact q_s times with the signer, playing the interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S},\mathcal{A}\rangle$:
  - $\mathcal{A}$ sends the bit-per-bit encryptions c_i for $i \in \{1,\ldots,\ell\}$, and the extra ciphertext (d_0, d_1) hiding Y the verification key vk_1 raised to the randomness;
  - Thanks to dk, $\mathcal{B}$ is able to extract M from the bit-per-bit ciphertexts (either the decryption leads to u_i and so $M_i = 1$, or to g_1 and so $M_i = 0$), and $Y = \mathsf{vk}_1^r$ from the additional ciphertext (d_0, d_1). One can also compute $c_0^{1/\gamma} = g_s^r$.
  - If one of the extracted terms is not of the right form (either not a bit in the c_i, or $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is not a Diffie-Hellman tuple, which occurs if $e(g_s^r, \mathsf{vk}_2) \neq e(Y, g_2)$ and can thus be checked with a pairing computation), then $\mathcal{A}$ has submitted a "word" not in the appropriate language for the SPHF. Therefore through the smoothness property of the SPHF, it is impossible from a theoretic point of view that the adversary extracts anything from $\mathcal{B}$'s answer, therefore $\mathcal{B}$ simply sends a random element Σ in $\mathbb{G}_1$ together with a valid random pair (g_1^t, g_2^t).
  - If $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is a Diffie-Hellman tuple, one knows that $Y = \mathsf{vk}_1^r$.
    $\mathcal{B}$ computes $H = -2jq_s + y_0 + \sum y_i M_i$ and $J = z_0 + \sum z_i M_i$, $\mathcal{F}(M) = h_s^H g_s^J$. If $H \equiv 0 \mod p$, it aborts, else it sets
    $$\sigma = (\mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s, (\mathsf{vk}_1^{-1/H} g_1^s, \mathsf{vk}_2^{-1/H} g_2^s)),$$
    for some random scalar s. Setting $t = s - x/H$, we can see this is indeed a valid signature (as output as the end of the signing interactive protocol), since we have:
    $$\begin{aligned}
    \sigma_1 &= \mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s = \mathsf{vk}_1^{-J/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^s \\
    &= g_s^{-xJ/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^t (h_s^H g_s^J g_s^r)^{x/H} = h^x (h^H g_s^J g_s^r)^t \\
    &= \mathsf{sk} \cdot \delta^t \qquad \text{where } \delta = \mathcal{F}(M) \times g_s^r \\
    \sigma_{2,1} &= \mathsf{vk}_1^{-1/H} g_1^s = g_1^{-x/H} g_1^s = g_1^t \\
    \sigma_{2,2} &= \mathsf{vk}_2^{-1/H} g_2^s = g_2^{-x/H} g_2^s = g_2^t
    \end{aligned}$$
  - $\mathcal{B}$ then acts honestly to send the signature through the SPHF.

  After a q_s queries, $\mathcal{A}$ outputs a valid signature σ^* on a new message M^* with non negligible probability.
- As before $\mathcal{B}$ computes $H^* = -2jq_s + y_0 + \sum y_i M_i^*$ and $J^* = z_0 + \sum z_i M_i^*$, $\mathcal{F}(M) = h^{H^*} g_1^{J^*}$.
- If $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ aborts. Otherwise $\sigma^* = (\mathsf{sk} \cdot \mathcal{F}(M^*)^t, g_s^t, g_2^t) = (\mathsf{sk} \cdot g_s^{tJ^*}, g_s^t, g_2^t)$ and so $\sigma_1^*/{\sigma_2^*}^{J^*} = \mathsf{sk} = h_s^x$. Therefore if $\mathcal{A}$'s signature is valid and if $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ solves its CDH^+ challenge.

The probability that all the $H \not\equiv 0 \mod p$ for all the simulations, but $H^* \equiv 0 \mod p$ in the forgery is the $(1, q_s)$-programmability of the Waters function. A full proof showing that it happens with probability in $\Theta(\mathsf{Succ}^{\mathsf{CDH}}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})/q_s\sqrt{\ell})$ can be found in [HK08]. ∎

I.D Generic Framework for SPHFs and New Constructions

In this appendix, we introduce our full generic framework for SPHFs using a new notion of graded rings, derived from [GGH13a]. It enables to deal with cyclic groups, bilinear groups

(with symmetric or asymmetric pairings), or even groups with multi-linear maps. Namely, it handles all the previous constructions from [BBC+13a].

Before introducing graded rings and our generic framework, we briefly recall the definition of bilinear groups. The last three subsections are dedicated to instantiations. The last instantiation can deal with any quadratic pairing product equation over ciphertexts, which encompasse all languages handled by Groth-Sahai NIZKs, and so can deal with any NP language. We can see that our generic scheme greatly simplify the construction and the presentation of all the SPHFs presented in these last subsections.

This appendix is very formal and technical. We strongly recommend the reader to first read Sections I.D.3 and I.4 where we give the intuition.

I.D.1 Bilinear Groups

Let us consider three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ or $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is called a *bilinear setting* if $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map (called a pairing) with the following properties:

- *Bilinearity.* For all $(a, b) \in \mathbb{Z}_p^2$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- *Non-degeneracy.* The element $e(g_1, g_2)$ generates $\mathbb{G}_T$;
- *Efficient computability.* The function e is efficiently computable.

It is called a *symmetric* bilinear setting if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. In this case, we denote it $(p, \mathbb{G}, \mathbb{G}_T, e)$ and we suppose $g = g_1 = g_2$. Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an *asymmetric* bilinear setting one otherwise.

I.D.2 Graded Rings

Our graded rings are a practical way to manipulate elements of various groups involved with pairings, and more generally, with multi-linear maps. This is a slight variant of the notion of graded encoding proposed in [GGH13a], where each element has only one representation, instead of a set of representations, and where we can add two elements even with different indexes.

Indexes Set.

As in [GGH13a], let us consider a finite set of indexes $\Lambda = \{0, \ldots, \kappa\}^\tau \subset \mathbb{N}^\tau$. In addition to considering the addition law $+$ over Λ, we also consider Λ as a bounded lattice, with the two following laws:

$$\sup(\vec{v}, \vec{v}') = (\max(\vec{v}_1, \vec{v}'_1), \ldots, \max(\vec{v}_\tau, \vec{v}'_\tau)) \qquad \inf(\vec{v}, \vec{v}') = (\min(\vec{v}_1, \vec{v}'_1), \ldots, \min(\vec{v}_\tau, \vec{v}'_\tau)).$$

We also write $\vec{v} < \vec{v}'$ (resp. $\vec{v} \leq \vec{v}'$) if and only if for all $i \in \{1, \ldots, \tau\}$, $\vec{v}_i < \vec{v}'_i$ (resp. $\vec{v}_i \leq \vec{v}'_i$). Let $\bar{0} = (0, \ldots, 0)$ and $\top = (\kappa, \ldots, \kappa)$, be the minimal and maximal elements.

Graded Ring.

The (κ, τ)-graded ring for a commutative ring R is the set $\mathfrak{G} = \Lambda \times R = \{[\vec{v}, x] \mid \vec{v} \in \Lambda, x \in R\}$, where $\Lambda = \{0, \ldots, \kappa\}^\tau$, with two binary operations $(+, \cdot)$ defined as follows:

- for every $u_1 = [\vec{v}_1, x_1], u_2 = [\vec{v}_2, x_2] \in \mathfrak{G}$: $u_1 + u_2 \stackrel{\text{def}}{=} [\sup(\vec{v}_1, \vec{v}_2), x_1 + x_2]$;
- for every $u_1 = [\vec{v}_1, x_1], u_2 = [\vec{v}_2, x_2] \in \mathfrak{G}$: $u_1 \cdot u_2 \stackrel{\text{def}}{=} [\vec{v}_1 + \vec{v}_2, x_1 \cdot x_2]$ if $\vec{v}_1 + \vec{v}_2 \in \Lambda$, or $\perp$ otherwise, where $\perp$ means the operation is undefined and cannot be done.

We remark that $\cdot$ is only a partial binary operation and we use the following convention: $\perp + u = u + \perp = u \cdot \perp = \perp \cdot u = \perp$, for any $u \in \mathfrak{G} \cup \{\perp\}$. We then denote $\mathfrak{G}_{\vec{v}}$ the additive group $\{u = [\vec{v}', x] \in \mathfrak{G} \mid \vec{v}' = \vec{v}\}$. We will make natural use of vector and matrix operations over graded ring elements.

Cyclic Groups and Pairing-Friendly Settings.

In the sequel, we consider graded rings over $R = \mathbb{Z}_p$ only, because we will use the vectorial space structure over $\mathbb{Z}_p$ in the proof of the smoothness of our generic construction of SPHF (see Section I.D.3). This means we cannot directly deal with constructions in [GGH13a] yet. Nevertheless, graded rings enable to easily deal with cyclic groups $\mathbb{G}$ of prime order p, and bilinear groups.

Cyclic Group In this case, $\kappa = \tau = 1$: elements $[0, x]$ of index 0 correspond to scalars $x \in \mathbb{Z}_p$ and elements $[1, x]$ of index 1 correspond to group elements $g^x \in \mathbb{G}$.

Symmetric Bilinear Group. Let $(p, \mathbb{G}, \mathbb{G}_T, e)$ be a symmetric bilinear group, and g be a generator of $\mathbb{G}$. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 2$ and $\tau = 1$. More precisely, we can consider the following map: $[0, x]$ corresponds to $x \in \mathbb{Z}_p$, $[1, x]$ corresponds to $g^x \in \mathbb{G}$ and $[2, x]$ corresponds to $e(g, g)^x \in \mathbb{G}_T$.

Asymmetric Bilinear Group. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be an asymmetric bilinear group, and g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 1$ and $\tau = 2$. More precisely, we can consider the following map: $[(0, 0), x]$ corresponds to $x \in \mathbb{Z}_p$, $[(1, 0), x]$ corresponds to $g_1^x \in \mathbb{G}_1$, $[(0, 1), x]$ corresponds to $g_2^x \in \mathbb{G}_2$ and $[(1, 1), x]$ corresponds to $e(g_1, g_2)^x \in \mathbb{G}_T$.

Notations. We have chosen an additive notation for the group law in $\mathfrak{G}_{\vec{v}}$. On the one hand, this a lot easier to write generic things done, but, on the other hand, it is a bit cumbersome for bilinear groups to use additive notations. Therefore, when we provide an example with a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, we use multiplicative notation $\cdot$ for the law in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and additive notation $+$ for the law in $\mathbb{Z}_p$, as soon as it is not too complicated. But when needed, we will also use the notation $\oplus$ and $\odot$ which correspond to the addition law and the multiplicative law of the corresponding graded rings. In other words, for any $x, y \in \mathbb{Z}_p$, $u_1, v_1 \in \mathbb{G}_1$, $u_2, v_2 \in \mathbb{G}_2$ and $u_T, v_T \in \mathbb{G}_T$, we have:

$$
\begin{array}{lll}
x \oplus y = x + y & & x \odot y = x \cdot y = xy \\
u_1 \oplus v_1 = u_1 \cdot v_1 = u_1 v_1 & & x \odot u_1 = u_1^x \\
u_2 \oplus v_2 = u_2 \cdot v_2 = u_2 v_2 & & x \odot u_1 = u_1^x \\
u_T \oplus v_T = u_T \cdot v_T & u_1 \odot u_2 = e(u_1, u_2) & x \odot u_T = u_T^x.
\end{array}
$$

The element 1 will always denote the neutral element in either $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ (depending on the context) and not $1 \in \mathbb{Z}_p$, which is not used in our constructions.

I.D.3 Generic Framework for GL-SPHF/KV-SPHF

In this section, we exhibit a generic framework for SPHF for languages of ciphertexts. This is an extension of the framework described in Section I.3 to graded rings. We assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \textsc{Lof}\mathrm{C}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$.

Language Representation.

For a language $\mathrm{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : \mathcal{S}et \mapsto \mathfrak{G}^{k\times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathfrak{G}^{1\times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \vec{\lambda} \in \mathfrak{G}^{1\times k}$ such that $\Theta_{\mathsf{aux}}(C) = \vec{\lambda} \cdot \Gamma(C))$. If Γ is a constant function (independent of the word C), this defines a KV-SPHF, otherwise this is a GL-SPHF. However, in any case, we need the indexes of the components of $\Gamma(C)$ to be independent of C.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute $\vec{\lambda}$.

Smooth Projective Hash Function.

With the above notations, the hashing key is a vector $\mathsf{hk} = \vec{\alpha} = (\alpha_1, \ldots, \alpha_n)^\intercal \xleftarrow{\$} \mathbb{Z}_p^n$, while the projection key is, for a word C, $\mathsf{hp} = \vec{\gamma}(C) = \Gamma(C) \cdot \vec{\alpha} \in \mathfrak{G}^k$ (if Γ does not depend on C, hp does not depend on C either). Then, the hash value is:

$$H = \mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\text{def}}{=} \quad \Theta_{\mathsf{aux}}(C) \cdot \vec{\alpha} = \vec{\lambda} \cdot \vec{\gamma}(C) \quad \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w) = H'.$$

The set Π of hash values is exactly $\mathfrak{G}_{\vec{v}_H}$, the set of graded elements of index $\vec{v}_H$, the maximal index of the elements of $\Theta_{\mathsf{aux}}(C)$.

In addition, the following security analysis proves that the above generic SPHF is perfectly smooth, and thus proves the Theorem I.2.2 as a particular case. We insist that if Γ really depends on C this construction yields a GL-SPHF, whereas when Γ is a constant matrix, we obtain a KV-SPHF, but perfectly smooth in both cases.

Security Analysis.

In order to prove the smoothness of the above SPHF, we consider a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \vec{\gamma}(C) = \Gamma(C) \cdot \vec{\alpha}$: $\forall \vec{\lambda} \in \mathfrak{G}^{1\times k}, \Theta_{\mathsf{aux}}(C) \neq \vec{\lambda} \cdot \Gamma(C)$. Using the projection $\mathfrak{L} : \mathfrak{G} \to \mathbb{Z}_p; u = [\vec{v}, x] \mapsto x$, which can be seen as the discrete logarithm, and which can be applied component-wise on vectors and matrices, this means that $\mathfrak{L}(\Theta_{\mathsf{aux}}(C))$ is linearly independent from the rows of $\mathfrak{L}(\Gamma(C))$. As a consequence, since $\vec{\alpha}$ is uniformly random, $\mathfrak{L}(\Theta_{\mathsf{aux}}(C)) \cdot \vec{\alpha}$ is a random variable independent from $\mathfrak{L}(\vec{\gamma}(C)) = \mathfrak{L}(\Gamma(C)) \cdot \vec{\alpha}$, and so from $\mathsf{hp} = \vec{\gamma}(C)$, since the index of $\vec{\gamma}(C)$ is a constant and thus $\mathfrak{L}(\vec{\gamma}(C))$ completely defines $\vec{\gamma}(C)$. Therefore, H is a uniform element of $\mathfrak{G}_{\vec{v}_H}$ given hp, aux and C.

I.D.4 Instantiations

A First Example with Pairings.

Notations. We consider the same kind of equation as in the body of the paper (Section I.4.1), but on possibly two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$, of the same prime order p, generated by g_1 and g_2, respectively, with a possible bilinear map into $\mathbb{G}_T$. We assume the DDH assumption hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$. We define ElGamal encryption schemes with encryption keys $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$ and $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ on each group. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i)$, for $X_1, \ldots, X_{n_1} \in \mathbb{G}_1$, and $C_{2,j} = (u_{2,j} = g_2^{r_{2,j}}, e_{2,j} = h_2^{r_{2,j}} \cdot g_2^{y_j})$, for $y_1, \ldots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{rcl} \mathsf{crs} & = & (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{ek}_1, \mathsf{ek}_2) \\ \mathsf{aux} & = & (a_1, \ldots, a_{n_1}, A_1, \ldots, A_{n_2}, B) \in \mathbb{Z}_p^{n_2} \times \mathbb{G}_1^{n_2+1}. \end{array} \tag{I.2}$$

We insist that here, contrarily to equation (I.1) in Section I.4.1, the group elements $(A_1, \ldots, A_{n_2})$ are part of aux, and thus not known in advance. The matrix Γ cannot depend on them anymore:

$$\Gamma = \left(\begin{array}{c|ccc|c} g_1 & 1 & \ldots & 1 & h_1 \\ \hline 1 & & & & h_2 \\ \vdots & & \mathbf{1} & & \vdots \\ 1 & & & & h_2 \end{array}\right)$$

and

$$\begin{aligned} \Theta_{\mathsf{aux}}(\vec{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right) \\ \vec{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, (A_j^{r_{2,j}})_j) \\ \vec{\lambda} \cdot \Gamma &= \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_{2,j}}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \textstyle\prod_j e(A_j^{r_{2,j}}, h_2)\right) \end{aligned}$$

We recall that in the matrix, 0 means $[\vec{v}, 0]$ for the appropriate index $\vec{v}$, and thus $1_{\mathbb{G}_1} = g_1^0 \in \mathbb{G}_1$ in the first line and column, but $1_{\mathbb{G}_2} = g_2^0 \in \mathbb{G}_2$ in the diagonal block. In addition, in the product $\vec{\lambda} \cdot \Gamma$, when adding two elements, they are first lifted in the minimal common higher ring, and when multiplying two elements, we either make a simple exponentiation (scalar with a group element) or a pairing (two group elements from different groups).

Because of the diagonal blocks in Γ, $\vec{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\vec{C})$, then the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\vec{C})$ is $\prod_i e(h_1^{r_{1,i}a_i} X^{a_i}, g_2) \cdot \prod_j e(A_j, h_2^{r_{2,j}} g_2^{y_j})/e(B, g_2)$, which is equal to the last component of $\vec{\lambda} \cdot \Gamma$, multiplied by the expression below, that is equal to 1 if and only if the relation (I.2) is satisfied:

$$\textstyle\prod_i e(X^{a_i}, g_2) \cdot \prod_j e(A_j, g_2^{y_j})/e(B, g_2) = e\left(\prod_i X^{a_i} \cdot \prod_j A_j^{y_j}/B, g_2\right).$$

It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g_1^\nu h_1^\lambda$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_2^\lambda)_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$H = \textstyle\prod_i e((u_{1,i}^\nu e_{1,i}^\lambda)^{a_i}, g_2) \cdot \prod_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^\lambda) \cdot e(B^{-\lambda}, g_2) = e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_{2,j}}, \mathsf{hp}_{2,j}) = H'.$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 1$ elements from $\mathbb{G}_1$ and $3n_2$ elements from $\mathbb{G}_2$, and pairings are required for the hash value.

Ciphertexts with Randomness Reuse. We can apply the same improvement as in Section I.4.1 by using multiple independent encryption keys in $\mathbb{G}_2$, $\mathsf{ek}_{2,j} = (g_2, h_{2,j} = g_2^{x_{2,j}})$, for $j = 1, \ldots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}_1^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g_2^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g_2^{y_j})_j)$, for $(y_j)_j \in \mathbb{Z}_p^{n_2}$, with $r_2 \in \mathbb{Z}_p$, still satisfying the same relation (I.2). This improves on the length of the ciphertexts of the g^{y_i}'s, from $2n_2$ group elements in $\mathbb{G}_2$ to $n_2 + 1$ in $\mathbb{G}_2$. A similar $\mathsf{KV\text{-}SPHF}$ as before can be derived, just modifying the last column vector $(h_2)_j$ by $(h_{2,j})_j$:

$$\Gamma = \left(\begin{array}{c|ccc|c} g_1 & 1 & \ldots & 1 & h_1 \\ \hline 1 & & & & h_{2,1} \\ \vdots & & \mathbf{1} & & \vdots \\ 1 & & & & h_{2,n_2} \end{array}\right)$$

and

$$\begin{aligned}\Theta_{\mathsf{aux}}(\vec{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right)\\ \vec{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, (A_j^{r_2})_j)\\ \vec{\lambda}\cdot\Gamma &= \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_2}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \textstyle\prod_j e(A_j^{r_2}, h_{2,j})\right)\end{aligned}$$

It leads to the following KV-SPHF, with $\mathsf{hp}_1 = g_1^{\nu} h_1^{\lambda}$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_{2,j}^{\lambda})_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$H = \textstyle\prod_i e((u_{1,i}^{\nu} e_{1,i}^{\lambda})^{a_i}, g_2) \cdot \prod_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^{\lambda}) \cdot e(B^{-\lambda}, g_2) = e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_2}, \mathsf{hp}_{2,j}) = H'.$$

Globally, the ciphertexts and the projection keys consist of $2n_1+1$ elements from $\mathbb{G}_1$ and $2n_2+1$ elements from $\mathbb{G}_2$, but pairings are still required for the hash value. The prior knowledge of the A_j's allows to avoid pairings, as shown in Section I.4.1.

SPHF for Linear Pairing Equations over Ciphertexts.

Let us now construct an KV-SPHF for a linear pairing equation in an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$ over ElGamal commitments. This will actually be a particular case of the construction of the next section for quadratic pairing equation. It is thus a warm-up for this more technical instantiation. The construction can obviously be extended to systems of linear pairing equations, and to other commitments schemes using the same methods as in Section I.4. It can also be slightly simplified in the case of symmetric bilinear groups.

Notations. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a (asymmetric) bilinear group. Let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and let $g_T = e(g_1, g_2)$. Let $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$, $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ and $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ be ElGamal key for encryption scheme in, respectively, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\textstyle\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{I.3}$$

with $\mathsf{aux} = ((A_{1,j})_j, (A_{2,i})_i, (a_{T,k})_k) \in \mathbb{G}_1^{n_2} \times \mathbb{G}_2^{n_1} \times \mathbb{Z}_p^{n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1, 2, T\}$ and $\iota \in \{1, \ldots, n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$. Words of $\mathcal{S}et$ are tuple $C = (C_{\omega,\iota})_{\omega\in\{1,2,T\},\, \iota\in\{1,\ldots,n_\omega\}}$.

Basic Scheme in $\mathbb{G}_T$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 & 1 & 1 & h_1 \\ 1 & g_2 & 1 & h_2 \\ 1 & 1 & g_T & h_T \end{pmatrix} \quad \Theta(C) = \begin{pmatrix} \bigoplus_i A_{2,i} \odot u_{1,i}, \bigoplus_j A_{1,j} \odot u_{2,j}, \bigoplus_k a_{T,k} \odot u_{T,k}, \\ \qquad (\bigoplus_i A_{2,i} \odot e_{1,i}) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus (\bigoplus_k a_{T,k} \odot e_{T,k}) \ominus B \end{pmatrix}.$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\vec{\lambda} = (\bigoplus_i A_{2,i} \odot r_{1,i}, \bigoplus_j A_{1,j} \odot r_{2,j}, \bigoplus_k r_{T,k}) = (\prod_i A_{2,i}^{r_{1,i}}, \prod_j A_{1,j}^{r_{2,j}}, \sum_k r_{T,k}).$$

We then have $\vec{\lambda} \odot \Gamma = \Theta(C)$ if and only if

$$\prod_i e(h_1^{r_{1,i}}, A_{2,i}) \cdot \prod_j e(A_{1,j}, h_2^{r_{2,j}}) \cdot \prod_k h_T^{r_{T,k}} = \prod_i e(e_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, e_{2,j}) \cdot \prod_k e_{T,k}^{a_{T,k}} / B$$

and thus if and only if Equation (I.3) is true, *i.e.*, the word is in the language. Furthermore, if we set $\gamma_1 = g_1^{\alpha_1} h_1^{\alpha^4}$, $\gamma_2 = g_2^{\alpha_2} h_2^{\alpha^4}$, and $\gamma_3 = g_T^{\alpha_3} h_T^{\alpha^4}$, we have

$$\begin{aligned} H &= \left(\prod_{i=1}^{n_1} e(u_{1,i}, A_{2,i})\right)^{\alpha_1} \cdot \left(\prod_{j=1}^{n_2} e(A_{1,j}, u_{2,j})\right)^{\alpha_2} \cdot \left(\prod_{k=1}^{n_T} u_{T,k}^{a_{T,k}}\right)^{\alpha_3} \\ &\times \left(\prod_{i=1}^{n_1} e(e_{1,i}, A_{2,i}) \cdot \prod_{j=1}^{n_2} e(A_{1,j}, e_{2,j}) \cdot \prod_{k=1}^{n_T} e_{T,k}^{a_{T,k}} / B\right)^{\alpha_4} \\ &= e(\gamma_1, \textstyle\prod_i A_{2,i}^{r_{1,i}}) \cdot e(\prod_j A_{1,j}^{r_{2,j}}, \gamma_2) \cdot \gamma_3^{\sum_k r_{T,k}} = H'. \end{aligned}$$

Variant. The above scheme is not efficient enough for practical use because elements in $\mathbb{G}_T$ are often big and operations in $\mathbb{G}_T$ are often slow. If $h_T = e(h_1, g_2)$, then the last row of Γ can be $(0, 0, g_1, h_1)$ which enables faster hashing and shorter projection key. We remark this modified encryption scheme in $\mathbb{G}_T$ is IND-CPA as soon as DDH is hard in $\mathbb{G}_1$, which we need to suppose for the ElGamal encryption scheme in $\mathbb{G}_1$ to be IND-CPA. So this variant is always more efficient when using ElGamal encryption.

However, if DDH is easy, as in symmetric bilinear group, this variant may not be interesting, since it requires to use the linear encryption scheme in $\mathbb{G}_T$ instead of the ElGamal one.

SPHF for Quadratic Pairing Equations over Ciphertexts.

In this section, we present a KV-SPHF for language of ElGamal commitments verifying a quadratic pairing equation. As usual, it can be extended to systems of quadratic pairing equations, and to other commitments schemes. We use the same notations as in the previous construction.

Example. Before showing the generic construction, we describe it on a simple example: we are interested in languages of the ciphertexts $C_1 = (u_1 = g_1^{r_1}, e_1 = h_1^{r_1} X_1)$ and $C_2 = (u_2 = g_2^{r_2}, e_2 = h_2^{r_2} X_2)$, that encrypt two values X_1 and X_2 such that $e(X_1, X_2) = B$ where B is some constant in $\mathbb{G}_T$ and $\mathsf{aux} = B$. We remark the equation $e(X_1, X_2) = B$ can also be written $X_1 \odot X_2 = B$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & h_1 \\ 1 & 1 & g_2 & h_2 \end{pmatrix} \quad \begin{aligned} \Theta(C) &= (-u_1 \odot u_2, u_1 \odot e_2, e_1 \odot u_2, e_1 \odot e_2 \ominus B) \\ &= (e(u_1, u_2)^{-1}, e(u_1, e_2), e(e_1, u_2), e(e_1, e_2)/B). \end{aligned}$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\vec{\lambda} = (-r_1 r_2, r_1 \odot e_2, r_2 \odot e_1) = (-r_1 r_2, e_2^{r_1}, e_1^{r_2}).$$

We have $\vec{\lambda} \odot \Gamma = \Theta(C)$ if and only if $e(h_1, h_2)^{-r_1 r_2} \cdot e(h_1, e_2^{r_1}) \cdot e(e_1^{r_2}, h_2) = e(e_1, e_2)/B$, and thus,

$$\begin{aligned} B &= e(e_1, e_2)/(e(h_1^{r_1}, X_2) \cdot e(e_1, h_2^{r_2})) \\ &= e(e_1, X_2)/e(h_1^{r_1}, X_2) = e(X_1, X_2) \end{aligned}$$

For the sake of completeness, if $\gamma_1 = e(g_1, g_2)^{\alpha_1} e(h_1, h_2)^{\alpha_4}$, $\gamma_2 = g_1^{\alpha_2} h_1^{\alpha_4}$, and $\gamma_3 = g_2^{\alpha_3} h_2^{\alpha_4}$, the corresponding hash value is:

$$H = e(u_1, u_2)^{-\alpha_1} \cdot e(u_1, e_2)^{\alpha_2} \cdot e(e_1, u_2)^{\alpha_3} \cdot (e(e_1, e_2)/B)^{\alpha_4} = \gamma_1^{-r_1 r_2} \cdot e(\gamma_2, e_2^{r_1}) \cdot e(e_1^{r_2}, \gamma_3).$$

Notations. Let us now introduce notation to handle any quadratic equation. In addition to previous notations, as in Section I.D.4, we also write $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ a public key for ElGamal encryption scheme in $\mathbb{G}_T$. We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_i \prod_j e(X_{1,i}, X_{2,j})^{a_{i,j}} \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{I.4}$$

with $\mathsf{aux} = ((A_{2,i})_i, (A_{1,j})_j, (a_{i,j})_{i,j}, (a_{T,k})_k) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{Z}_p^{n_1 n_2 + n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{i=1}^{n_1} \bigoplus_{j=1}^{n_2} a_{i,j} \odot X_{1,i} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1, 2, T\}$ and $\iota \in \{1, \ldots, n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$.

Basic Scheme in $\mathbb{G}_T$. Let us consider the following matrix, with a diagonal block

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & 1 & h_1 \\ 1 & 1 & g_2 & 1 & h_2 \\ 1 & 1 & 1 & g_T & h_T \end{pmatrix}$$

With

$$\Theta(C) = \begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot u_{1,i} \odot u_{2,j}, \left(\bigoplus_i \bigoplus_j a_{i,j} \odot u_{1,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot u_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{1,i} \odot u_{2,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot u_{2,j}\right), \bigoplus_i a_{T,i} \odot u_{T,i}, \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{2,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot e_{1,i}\right) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_k a_{T,k} \odot e_{T,k}\right) \ominus B \end{pmatrix}$$

the requirement $\vec{\lambda} \odot \Gamma = \Theta(C)$ implies

$$\begin{aligned} \vec{\lambda} &= \begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot r_{1,i} \odot r_{2,j}, \left(\bigoplus_i \bigoplus_j r_{1,i} \odot a_{i,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot r_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j r_{2,i} \odot a_{i,j} \odot e_{1,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot r_{2,j}\right), \bigoplus_k r_{T,k} \end{pmatrix} \\ &= \left(\sum_i \sum_j a_{i,j} r_{1,i} r_{2,j}, \prod_i \prod_j e_{2,j}^{r_{1,i} a_{i,j}} \cdot \prod_i A_{2,i}^{r_{1,i}}, \prod_i \prod_j e_{1,j}^{r_{2,i} a_{i,j}} \cdot \prod_j A_{1,j}^{r_{2,j}}, \sum_k r_{T,k} \right), \end{aligned}$$

and it is satisfied, if and only if Equation (I.4) is true, *i.e.*, the word is in the language.

Variant. The same trick as the one used in the variant of the SPHF for linear pairing equation can be used to avoid having too many elements of the projection key in $\mathbb{G}_T$.

Bibliography

[ABB+10] José Bacelar Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider, *A certifying compiler for zero-knowledge proofs of knowledge based on sigma-protocols*, ESORICS 2010: 15th European Symposium on Research in Computer Security (Athens, Greece) (Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, eds.), Lecture Notes in Computer Science, vol. 6345, Springer, Berlin, Germany, September 20–22, 2010, pp. 151–167. (Cited on page 55.)

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval, *SPHF-friendly non-interactive commitments*, Advances in Cryptology – ASIACRYPT 2013, Part I (Bengalore, India) (Kazue Sako and Palash Sarkar, eds.), Lecture Notes in Computer Science, vol. 8269, Springer, Berlin, Germany, December 1–5, 2013, pp. 214–234. (Cited on page 56.)

[ABC+11] Michel Abdalla, James Birkett, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, Jacob C. N. Schuldt, and Nigel P. Smart, *Wildcarded identity-based encryption*, Journal of Cryptology **24** (2011), no. 1, 42–82. (Cited on page 24, 25.)

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven, *Robust encryption*, TCC 2010: 7th Theory of Cryptography Conference (Zurich, Switzerland) (Daniele Micciancio, ed.), Lecture Notes in Computer Science, vol. 5978, Springer, Berlin, Germany, February 9–11, 2010, pp. 480–497. (Cited on page 18.)

[ACGP11] Michel Abdalla, Céline Chevalier, Louis Granboulan, and David Pointcheval, *Contributory password-authenticated group key exchange with join capability*, Topics in Cryptology – CT-RSA 2011 (San Francisco, CA, USA) (Aggelos Kiayias, ed.), Lecture Notes in Computer Science, vol. 6558, Springer, Berlin, Germany, February 14–18, 2011, pp. 142–160. (Cited on page 280.)

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros, *Practical group signatures without random oracles*, Cryptology ePrint Archive, Report 2005/385, 2005, `http://eprint.iacr.org/2005/385`. (Cited on page 5, 32, 196.)

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik, *A practical and provably secure coalition-resistant group signature scheme*, Advances in Cryptology – CRYPTO 2000 (Santa Barbara, CA, USA) (Mihir Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer, Berlin, Germany, August 20–24, 2000, pp. 255–270. (Cited on page 32, 196.)

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval, *Smooth projective hashing for conditionally extractable commitments*, Advances in Cryptology – CRYPTO 2009 (Santa Barbara, CA, USA) (Shai Halevi, ed.), Lecture Notes in Computer Science, vol. 5677, Springer, Berlin, Germany, August 16–20, 2009, pp. 671–689. (Cited on page 41, 46, 50, 214, 215, 216, 226, 229, 230, 235, 243, 248, 249, 251, 253, 256, 258, 279, 288, 294.)

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin, *On the security of joint signature and encryption*, Advances in Cryptology – EUROCRYPT 2002 (Amsterdam, The Netherlands) (Lars R. Knudsen, ed.), Lecture Notes in Computer Science, vol. 2332, Springer, Berlin, Germany, April 28 – May 2, 2002, pp. 83–107. (Cited on page 75, 76, 95.)

[AF96] Masayuki Abe and Eiichiro Fujisaki, *How to date blind signatures*, Advances in Cryptology – ASIACRYPT'96 (Kyongju, Korea) (Kwangjo Kim and Tsutomu Matsumoto, eds.), Lecture Notes in Computer Science, vol. 1163, Springer, Berlin, Germany, November 3–7, 1996, pp. 244–251. (Cited on page 39, 150, 178.)

[AF07] Masayuki Abe and Serge Fehr, *Perfect NIZK with adaptive soundness*, TCC 2007: 4th Theory of Cryptography Conference (Amsterdam, The Netherlands) (Salil P. Vadhan, ed.), Lecture Notes in Computer Science, vol. 4392, Springer, Berlin, Germany, February 21–24, 2007, pp. 118–136. (Cited on page 94.)

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo, *Structure-preserving signatures and commitments to group elements*, Advances in Cryptology – CRYPTO 2010 (Santa Barbara, CA, USA) (Tal Rabin, ed.), Lecture Notes in Computer Science, vol. 6223, Springer, Berlin, Germany, August 15–19, 2010, pp. 209–236. (Cited on page 37, 39, 150, 213, 214.)

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger, *Improved proxy re-encryption schemes with applications to secure distributed storage*, ISOC Network and Distributed System Security Symposium – NDSS 2005 (San Diego, California, USA), The Internet Society, February 3–4, 2005. (Cited on page 23.)

[AFGH06] ———, *Improved proxy re-encryption schemes with applications to secure distributed storage*, ACM Trans. Inf. Syst. Secur. **9** (2006), no. 1, 1–30. (Cited on page 23, 24, 70, 71, 72, 74, 76, 78, 83, 87, 92.)

[AH05] Giuseppe Ateniese and Susan Hohenberger, *Proxy re-signatures: New definitions, algorithms, and applications*, ACM CCS 05: 12th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, eds.), ACM Press, November 7–11, 2005, pp. 310–319. (Cited on page 25, 26, 87, 88, 89, 90, 91, 93, 94, 95, 100, 103.)

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton, *Secret handshakes with dynamic and fuzzy matching*, ISOC Network and Distributed System Security Symposium – NDSS 2007 (San Diego, California, USA), The Internet Society, February 28 – March 2, 2007. (Cited on page 51, 248, 261.)

[AL10] Yonatan Aumann and Yehuda Lindell, *Security against covert adversaries: Efficient protocols for realistic adversaries*, Journal of Cryptology **23** (2010), no. 2, 281–343. (Cited on page 236, 307.)

[AMO08] Norio Akagi, Yoshifumi Manabe, and Tatsuaki Okamoto, *An efficient anonymous credential system*, FC 2008: 12th International Conference on Financial Cryptography and Data Security (Cozumel, Mexico) (Gene Tsudik, ed.), Lecture Notes in Computer Science, vol. 5143, Springer, Berlin, Germany, January 28–31, 2008, pp. 272–286. (Cited on page 35.)

[AO00] Masayuki Abe and Tatsuaki Okamoto, *Provably secure partially blind signatures*, Advances in Cryptology – CRYPTO 2000 (Santa Barbara, CA, USA) (Mihir Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer, Berlin, Germany, August 20–24, 2000, pp. 271–286. (Cited on page 39, 150.)

[AO01] Masayuki Abe and Miyako Ohkubo, *Provably secure fair blind signatures with tight revocation*, Advances in Cryptology – ASIACRYPT 2001 (Gold Coast, Australia) (Colin Boyd, ed.), Lecture Notes in Computer Science, vol. 2248, Springer, Berlin, Germany, December 9–13, 2001, pp. 583–602. (Cited on page 39, 178.)

[AWSM07] Man Ho Au, Qianhong Wu, Willy Susilo, and Yi Mu, *Compact e-cash from bounded accumulator*, Topics in Cryptology – CT-RSA 2007 (San Francisco, CA, USA) (Masayuki Abe, ed.), Lecture Notes in Computer Science, vol. 4377, Springer, Berlin, Germany, February 5–9, 2007, pp. 178–195. (Cited on page 35.)

[BB04a] Dan Boneh and Xavier Boyen, *Efficient selective-ID secure identity based encryption without random oracles*, Advances in Cryptology – EUROCRYPT 2004 (Interlaken, Switzerland) (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science, vol. 3027, Springer, Berlin, Germany, May 2–6, 2004, pp. 223–238. (Cited on page 75, 76, 84, 101, 128, 170, 210.)

[BB04b] ———, *Short signatures without random oracles*, Advances in Cryptology – EUROCRYPT 2004 (Interlaken, Switzerland) (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science, vol. 3027, Springer, Berlin, Germany, May 2–6, 2004, pp. 56–73. (Cited on page 32, 182, 196, 200.)

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud, *Efficient UC-secure authenticated key-exchange for algebraic languages*, PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography (Nara, Japan) (Kaoru Kurosawa and Goichiro Hanaoka, eds.), Lecture Notes in Computer Science, vol. 7778, Springer, Berlin, Germany, February 26 – March 1, 2013, pp. 272–291. (Cited on page 41, 46, 51, 52, 54, 247, 295, 296, 298, 300, 301, 304, 305, 319.)

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud, *New techniques for SPHFs and efficient one-round PAKE protocols*, Advances in Cryptology – CRYPTO 2013, Part I (Santa Barbara, CA, USA) (Ran Canetti and Juan A. Garay, eds.), Lecture Notes in Computer Science, vol. 8042, Springer, Berlin, Germany, August 18–22, 2013, pp. 449–475. (Cited on page 41, 42, 45, 46, 52, 54, 293.)

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss, *Divertible protocols and atomic proxy cryptography*, Advances in Cryptology – EUROCRYPT'98 (Espoo, Finland) (Kaisa Nyberg, ed.), Lecture Notes in Computer Science, vol. 1403, Springer, Berlin, Germany, May 31 – June 4, 1998, pp. 127–144. (Cited on page 22, 23, 25, 26, 69, 70, 87, 88.)

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon, *Randomness re-use in multi-recipient encryption schemeas*, PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography (Miami, USA) (Yvo Desmedt, ed.), Lecture Notes in Computer Science, vol. 2567, Springer, Berlin, Germany, January 6–8, 2003, pp. 85–99. (Cited on page 302, 317, 322.)

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham, *Short group signatures*, Advances in Cryptology – CRYPTO 2004 (Santa Barbara, CA, USA) (Matthew Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer, Berlin, Germany, August 15–19, 2004, pp. 41–55. (Cited on page 5, 32, 51, 118, 140, 150, 174, 182, 196, 223, 231, 265, 295.)

[BCDP13] Olivier Blazy, Céline Chevalier, Léo Ducas, and Jiaxin Pan, *Errorless smooth projective hash function based on LWE*, Cryptology ePrint Archive, Report 2013/821, 2013, `http://eprint.iacr.org/2013/821`. (Cited on page 56.)

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz, *Chosen-ciphertext security from identity-based encryption*, SIAM J. Comput. **36** (2007), no. 5, 1301–1328. (Cited on page 75.)

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya, *P-signatures and noninteractive anonymous credentials*, TCC 2008: 5th Theory of Cryptography Conference (San Francisco, CA, USA) (Ran Canetti, ed.), Lecture Notes in Computer Science, vol. 4948, Springer, Berlin, Germany, March 19–21, 2008, pp. 356–374. (Cited on page 35, 36, 56.)

[BCKL09] ———, *Compact e-cash and simulatable VRFs revisited*, PAIRING 2009: 3rd International Conference on Pairing-based Cryptography (Palo Alto, CA, USA) (Hovav Shacham and Brent Waters, eds.), Lecture Notes in Computer Science, vol. 5671, Springer, Berlin, Germany, August 12–14, 2009, pp. 114–131. (Cited on page 35, 36, 56.)

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin, *Secure computation without authentication*, Advances in Cryptology – CRYPTO 2005 (Santa Barbara, CA, USA) (Victor Shoup, ed.), Lecture Notes in Computer Science, vol. 3621, Springer, Berlin, Germany, August 14–18, 2005, pp. 361–377. (Cited on page 250, 255, 258, 279, 280.)

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud, *Analysis and improvement of Lindell's UC-secure commitment schemes*, ACNS 13: 11th International Conference on Applied Cryptography and Network Security (Banff, AB, Canada) (Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, eds.), Lecture Notes in Computer Science, vol. 7954, Springer, Berlin, Germany, June 25–28, 2013, pp. 534–551. (Cited on page 41, 47.)

[BCV14] Olivier Blazy, Céline Chevalier, and Damien Vergnaud, *Non-interactive zero-knowledge proofs of non-membership*, in submission, 2014. (Cited on page 53.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway, *A concrete security treatment of symmetric encryption*, 38th Annual Symposium on Foundations of Computer Science (Miami Beach, Florida), IEEE Computer Society Press, October 19–22, 1997, pp. 394–403. (Cited on page 221.)

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology – CRYPTO'98 (Santa Barbara, CA, USA) (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer, Berlin, Germany, August 23–27, 1998, pp. 26–45. (Cited on page 215.)

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong, *Secret handshakes from pairing-based key agreements*, 2003 IEEE Symposium on Security and Privacy (Berkeley, California, USA), IEEE Computer Society Press, May 11–14, 2003, pp. 180–196. (Cited on page 48, 51, 214, 248.)

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek, *Standard security does not imply security against selective-opening*, Advances in Cryptology – EUROCRYPT 2012 (Cambridge, UK) (David Pointcheval and Thomas Johansson, eds.), Lecture Notes in Computer Science, vol. 7237, Springer, Berlin, Germany, April 15–19, 2012, pp. 645–662. (Cited on page 109.)

[Bea97] Donald Beaver, *Plug and play encryption*, Advances in Cryptology – CRYPTO'97 (Santa Barbara, CA, USA) (Burton S. Kaliski Jr., ed.), Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, Germany, August 17–21, 1997, pp. 75–89. (Cited on page 108.)

[BF01] Dan Boneh and Matthew K. Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology – CRYPTO 2001 (Santa Barbara, CA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer, Berlin, Germany, August 19–23, 2001, pp. 213–229. (Cited on page 197, 201.)

[BF03] ———, *Identity-based encryption from the weil pairing*, SIAM J. Comput. **32** (2003), no. 3, 586–615. (Cited on page 22, 23, 24, 34, 48, 70.)

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud, *Batch Groth-Sahai*, ACNS 10: 8th International Conference on Applied Cryptography and Network Security (Beijing, China) (Jianying Zhou and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 6123, Springer, Berlin, Germany, June 22–25, 2010, pp. 218–235. (Cited on page 27, 31, 178.)

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali, *Proving security against chosen cyphertext attacks*, Advances in Cryptology – CRYPTO'88 (Santa Barbara, CA, USA) (Shafi Goldwasser, ed.), Lecture Notes in Computer Science, vol. 403, Springer, Berlin, Germany, August 21–25, 1988, pp. 256–268. (Cited on page 27.)

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill, *On notions of security for deterministic encryption, and efficient constructions without random oracles*, Advances in Cryptology – CRYPTO 2008 (Santa Barbara, CA, USA) (David Wagner, ed.), Lecture Notes in Computer Science, vol. 5157, Springer, Berlin, Germany, August 17–21, 2008, pp. 335–359. (Cited on page 126, 127, 139.)

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud, *Signatures on randomizable ciphertexts*, PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography (Taormina, Italy) (Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, eds.), Lecture Notes in Computer Science, vol. 6571, Springer, Berlin, Germany, March 6–9, 2011, pp. 403–422. (Cited on page 6, 9, 15, 20, 27, 37, 48, 49, 150, 151, 152, 153, 159, 160, 174, 213, 214, 215, 226, 227, 232, 245, 266, 309, 314.)

[BFPV13] ———, *Short blind signatures*, Journal of Computer Security **21** (2013), no. 5, 627–661. (Cited on page 15, 20, 27, 37, 39, 149.)

[BFPW07] Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi, *A closer look at PKI: Security and efficiency*, PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography (Beijing, China) (Tatsuaki Okamoto and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 4450, Springer, Berlin, Germany, April 16–20, 2007, pp. 458–475. (Cited on page 32.)

[BG13] Stephanie Bayer and Jens Groth, *Zero-knowledge argument for polynomial evaluation with application to blacklists*, Advances in Cryptology – EUROCRYPT 2013 (Athens, Greece) (Thomas Johansson and Phong Q. Nguyen, eds.), Lecture Notes in Computer Science, vol. 7881, Springer, Berlin, Germany, May 26–30, 2013, pp. 646–663. (Cited on page 53.)

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg, *Space-efficient identity based encryption without pairings*, 48th Annual Symposium on Foundations of Computer Science (Providence, USA), IEEE Computer Society Press, October 20–23, 2007, pp. 647–657. (Cited on page 85.)

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé, *A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic*, Advances in Cryptology – EUROCRYPT 2014, Lecture Notes in Computer Science, Springer, Berlin, Germany, 2014, pp. 1–16. (Cited on page 5.)

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham, *Aggregate and verifiably encrypted signatures from bilinear maps*, Advances in Cryptology – EUROCRYPT 2003 (Warsaw, Poland) (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer, Berlin, Germany, May 4–8, 2003, pp. 416–432. (Cited on page 21.)

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim, *Evaluating 2-DNF formulas on ciphertexts*, TCC 2005: 2nd Theory of Cryptography Conference (Cambridge, MA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 3378, Springer, Berlin, Germany, February 10–12, 2005, pp. 325–341. (Cited on page 15.)

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin, *Fast batch verification for modular exponentiation and digital signatures*, Advances in Cryptology – EUROCRYPT'98 (Espoo, Finland) (Kaisa Nyberg, ed.), Lecture Notes in Computer Science, vol. 1403, Springer, Berlin, Germany, May 31 – June 4, 1998, pp. 236–250. (Cited on page 31.)

[BH92] Donald Beaver and Stuart Haber, *Cryptographic protocols provably secure against dynamic adversaries*, Advances in Cryptology – EUROCRYPT'92 (Balatonfüred, Hungary) (Rainer A. Rueppel, ed.), Lecture Notes in Computer Science, vol. 658, Springer, Berlin, Germany, May 24–28, 1992, pp. 307–323. (Cited on page 108.)

[BHK12] Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski, *On definitions of selective opening security*, PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography (Darmstadt, Germany) (Marc Fischlin, Johannes Buchmann, and Mark Manulis, eds.), Lecture Notes in Computer Science, vol. 7293, Springer, Berlin, Germany, May 21–23, 2012, pp. 522–539. (Cited on page 109.)

[BHS04] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons, *Concealing complex policies with hidden credentials*, ACM CCS 04: 11th Conference on Computer and Communications Security (Washington D.C., USA) (Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, eds.), ACM Press, October 25–29, 2004, pp. 146–157. (Cited on page 48, 214.)

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek, *Possibility and impossibility results for encryption and commitment secure under selective opening*, Advances in Cryptology – EUROCRYPT 2009 (Cologne, Germany) (Antoine Joux, ed.), Lecture Notes in Computer Science, vol. 5479, Springer, Berlin, Germany, April 26–30, 2009, pp. 1–35. (Cited on page 16, 17, 108, 109, 110, 112, 113, 114, 116, 118, 125, 128, 134, 135, 137, 138, 139, 140.)

[BK05] Dan Boneh and Jonathan Katz, *Improved efficiency for CCA-secure cryptosystems built using identity-based encryption*, Topics in Cryptology – CT-RSA 2005 (San Francisco, CA, USA) (Alfred Menezes, ed.), Lecture Notes in Computer Science, vol. 3376, Springer, Berlin, Germany, February 14–18, 2005, pp. 87–103. (Cited on page 82, 121.)

[Ble98] Daniel Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1*, Advances in Cryptology – CRYPTO'98 (Santa Barbara, CA, USA) (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer, Berlin, Germany, August 23–27, 1998, pp. 1–12. (Cited on page 118.)

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham, *Short signatures from the Weil pairing*, Journal of Cryptology **17** (2004), no. 4, 297–319. (Cited on page 19, 26, 88, 89, 94, 97.)

[BM92] Steven M. Bellovin and Michael Merritt, *Encrypted key exchange: Password-based protocols secure against dictionary attacks*, 1992 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1992, pp. 72–84. (Cited on page 49, 248, 294.)

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud, *Separation results on the "one-more" computational problems*, Topics in Cryptology – CT-RSA 2008 (San Francisco, CA, USA) (Tal Malkin, ed.), Lecture Notes in Computer Science, vol. 4964, Springer, Berlin, Germany, April 7–11, 2008, pp. 71–87. (Cited on page 19.)

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi, *Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions*, Advances in Cryptology – EUROCRYPT 2003 (Warsaw, Poland) (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer, Berlin, Germany, May 4–8, 2003, pp. 614–629. (Cited on page 32, 34, 196.)

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters, *Direct chosen ciphertext security from identity-based techniques*, ACM CCS 05: 12th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, eds.), ACM Press, November 7–11, 2005, pp. 320–329. (Cited on page 76, 82.)

[BN06] Mihir Bellare and Gregory Neven, *Multi-signatures in the plain public-key model and a general forking lemma*, ACM CCS 06: 13th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, eds.), ACM Press, October 30 – November 3, 2006, pp. 390–399. (Cited on page 89, 90, 92.)

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko, *The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme*, Journal of Cryptology **16** (2003), no. 3, 185–215. (Cited on page 19.)

[Bol03] Alexandra Boldyreva, *Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme*, PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography (Miami, USA) (Yvo Desmedt, ed.), Lecture Notes in Computer Science, vol. 2567, Springer, Berlin, Germany, January 6–8, 2003, pp. 31–46. (Cited on page 19, 92.)

[Bon98] Dan Boneh, *The decision Diffie-Hellman problem*, Third Algorithmic Number Theory Symposium (ANTS), Lecture Notes in Computer Science, vol. 1423, Springer, Berlin, Germany, 1998, Invited paper. (Cited on page 5.)

[BP04a] Mihir Bellare and Adriana Palacio, *The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols*, Advances in Cryptology – CRYPTO 2004 (Santa Barbara, CA, USA) (Matthew Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer, Berlin, Germany, August 15–19, 2004, pp. 273–289. (Cited on page 94.)

[BP04b] ———, *Towards plaintext-aware public-key encryption without random oracles*, Advances in Cryptology – ASIACRYPT 2004 (Jeju Island, Korea) (Pil Joong Lee, ed.), Lecture Notes in Computer Science, vol. 3329, Springer, Berlin, Germany, December 5–9, 2004, pp. 48–62. (Cited on page 94.)

[BP13a] Fabrice Benhamouda and David Pointcheval, *Trapdoor smooth projective hash functions*, Cryptology ePrint Archive, Report 2013/341, 2013, `http://eprint.iacr.org/2013/341`. (Cited on page 56.)

[BP13b] ———, *Verifier-based password-authenticated key exchange: New models and constructions*, Cryptology ePrint Archive, Report 2013/833, 2013, `http://eprint.iacr.org/2013/833`. (Cited on page 56.)

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway, *Authenticated key exchange secure against dictionary attacks*, Advances in Cryptology – EUROCRYPT 2000 (Bruges, Belgium) (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer, Berlin, Germany, May 14–18, 2000, pp. 139–155. (Cited on page 49, 294, 304, 305, 311.)

[BPV12a] Olivier Blazy, David Pointcheval, and Damien Vergnaud, *Compact round-optimal partially-blind signatures*, SCN 12: 8th International Conference on Security in Communication Networks (Amalfi, Italy) (Ivan Visconti and Roberto De Prisco, eds.), Lecture Notes in Computer Science, vol. 7485, Springer, Berlin, Germany, September 5–7, 2012, pp. 95–112. (Cited on page 15, 20, 27, 39.)

[BPV12b] ———, *Round-optimal privacy-preserving protocols with smooth projective hash functions*, TCC 2012: 9th Theory of Cryptography Conference (Taormina, Sicily, Italy) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 7194, Springer, Berlin, Germany, March 19–21, 2012, pp. 94–111. (Cited on page 41, 48, 52, 150, 213, 251, 254, 261, 294, 296, 306, 309, 314.)

[BR93] Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, ACM CCS 93: 1st Conference on Computer and Communications Security (Fairfax, Virginia, USA) (V. Ashby, ed.), ACM Press, November 3–5, 1993, pp. 62–73. (Cited on page 4, 23, 25, 26, 32, 34, 36, 70, 88, 196, 197.)

[Bri03] Ernest F. Brickell, *An efficient protocol for anonymously providing assurance of the container of the private key*, Submission to the Trusted Computing Group. April, 2003, 2003. (Cited on page 33, 196.)

[BS98] Dan Boneh and James Shaw, *Collusion-secure fingerprinting for digital data*, IEEE Transactions on Information Theory **44** (1998), no. 5, 1897–1905. (Cited on page 25.)

[BS99] Mihir Bellare and Amit Sahai, *Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization*, Advances in Cryptology – CRYPTO'99 (Santa Barbara, CA, USA) (Michael J. Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer, Berlin, Germany, August 15–19, 1999, pp. 519–536. (Cited on page 15.)

[BS04] Dan Boneh and Hovav Shacham, *Group signatures with verifier-local revocation*, ACM CCS 04: 11th Conference on Computer and Communications Security (Washington D.C., USA) (Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, eds.), ACM Press, October 25–29, 2004, pp. 168–177. (Cited on page 33, 34, 195, 196, 197, 198.)

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang, *Foundations of group signatures: The case of dynamic groups*, Topics in Cryptology – CT-RSA 2005 (San Francisco, CA, USA) (Alfred Menezes, ed.), Lecture Notes in Computer Science, vol. 3376, Springer, Berlin, Germany, February 14–18, 2005, pp. 136–153. (Cited on page 32, 40, 178, 180, 181, 196.)

[BT94] Josh Cohen Benaloh and Dwight Tuinstra, *Receipt-free secret-ballot elections (extended abstract)*, 26th Annual ACM Symposium on Theory of Computing (Montréal, Québec, Canada), ACM Press, May 23–25, 1994, pp. 544–553. (Cited on page 151.)

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, 52nd Annual Symposium on Foundations of Computer Science (Palm Springs, California, USA) (Rafail Ostrovsky, ed.), IEEE Computer Society Press, October 22–25, 2011, pp. 97–106. (Cited on page 55.)

[BV11b] ———, *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, Advances in Cryptology – CRYPTO 2011 (Santa Barbara, CA, USA) (Phillip Rogaway, ed.), Lecture Notes in Computer Science, vol. 6841, Springer, Berlin, Germany, August 14–18, 2011, pp. 505–524. (Cited on page 55.)

[BVZ12] Aurélie Bauer, Damien Vergnaud, and Jean-Christophe Zapalowicz, *Inferring sequences produced by nonlinear pseudorandom number generators using Coppersmith's methods*, PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography (Darmstadt, Germany) (Marc Fischlin, Johannes Buchmann, and Mark Manulis, eds.), Lecture Notes in Computer Science, vol. 7293, Springer, Berlin, Germany, May 21–23, 2012, pp. 609–626. (Cited on page 3.)

[BW06a] Xavier Boyen and Brent Waters, *Anonymous hierarchical identity-based encryption (without random oracles)*, Advances in Cryptology – CRYPTO 2006 (Santa Barbara, CA, USA) (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer, Berlin, Germany, August 20–24, 2006, pp. 290–307. (Cited on page 75.)

[BW06b] ———, *Compact group signatures without random oracles*, Advances in Cryptology – EUROCRYPT 2006 (St. Petersburg, Russia) (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 4004, Springer, Berlin, Germany, May 28 – June 1, 2006, pp. 427–444. (Cited on page 32, 151, 196, 213.)

[BW07] ———, *Full-domain subgroup hiding and constant-size group signatures*, PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography (Beijing, China) (Tatsuaki Okamoto and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 4450, Springer, Berlin, Germany, April 16–20, 2007, pp. 1–15. (Cited on page 32, 33, 196, 197, 198, 200, 201, 202, 207, 208, 210, 213.)

[BWY11] Mihir Bellare, Brent Waters, and Scott Yilek, *Identity-based encryption secure against selective opening attack*, TCC 2011: 8th Theory of Cryptography Conference (Providence, RI, USA) (Yuval Ishai, ed.), Lecture Notes in Computer Science, vol. 6597, Springer, Berlin, Germany, March 28–30, 2011, pp. 235–252. (Cited on page 109, 111.)

[BY09] Mihir Bellare and Scott Yilek, *Encryption schemes secure under selective opening attack*, Cryptology ePrint Archive, Report 2009/101, 2009, `http://eprint.iacr.org/2009/101`. (Cited on page 139, 140.)

[Can01] Ran Canetti, *Universally composable security: A new paradigm for cryptographic protocols*, 42nd Annual Symposium on Foundations of Computer Science (Las Vegas, Nevada, USA), IEEE Computer Society Press, October 14–17, 2001, pp. 136–145. (Cited on page 11, 13, 250, 305.)

[CCGS10] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup, *Credential authenticated identification and key exchange*, Advances in Cryptology – CRYPTO 2010 (Santa Barbara, CA, USA) (Tal Rabin, ed.), Lecture Notes in Computer Science, vol. 6223, Springer, Berlin, Germany, August 15–19, 2010, pp. 255–276. (Cited on page 51, 248, 254, 260, 296.)

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat, *Efficient protocols for set membership and range proofs*, Advances in Cryptology – ASIACRYPT 2008 (Melbourne, Australia) (Josef Pieprzyk, ed.), Lecture Notes in Computer Science, vol. 5350, Springer, Berlin, Germany, December 7–11, 2008, pp. 234–252. (Cited on page 53.)

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky, *Deniable encryption*, Advances in Cryptology – CRYPTO'97 (Santa Barbara, CA, USA) (Burton S. Kaliski Jr., ed.), Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, Germany, August 17–21, 1997, pp. 90–104. (Cited on page 108.)

[CF01] Ran Canetti and Marc Fischlin, *Universally composable commitments*, Advances in Cryptology – CRYPTO 2001 (Santa Barbara, CA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer, Berlin, Germany, August 19–23, 2001, pp. 19–40. (Cited on page 45, 46.)

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor, *Adaptively secure multi-party computation*, 28th Annual ACM Symposium on Theory of Computing (Philadephia, Pennsylvania, USA), ACM Press, May 22–24, 1996, pp. 639–648. (Cited on page 108, 109.)

[CFNP00] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas, *Tracing traitors*, IEEE Transactions on Information Theory **46** (2000), no. 3, 893–910. (Cited on page 24.)

[CG07] Sébastien Canard and Aline Gouget, *Divisible e-cash systems can be truly anonymous*, Advances in Cryptology – EUROCRYPT 2007 (Barcelona, Spain) (Moni Naor, ed.), Lecture Notes in Computer Science, vol. 4515, Springer, Berlin, Germany, May 20–24, 2007, pp. 482–497. (Cited on page 35.)

[CG08a] Jan Camenisch and Thomas Groß, *Efficient attributes for anonymous credentials*, ACM CCS 08: 15th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Peng Ning, Paul F. Syverson, and Somesh Jha, eds.), ACM Press, October 27–31, 2008, pp. 345–356. (Cited on page 36, 53.)

[CG08b] Sébastien Canard and Aline Gouget, *Anonymity in transferable e-cash*, ACNS 08: 6th International Conference on Applied Cryptography and Network Security (New York, NY, USA) (Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 5037, Springer, Berlin, Germany, June 3–6, 2008, pp. 207–223. (Cited on page 35.)

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi, *The random oracle methodology, revisited*, Journal of the ACM **51** (2004), no. 4, 557–594. (Cited on page 4, 197.)

[CGT06] Sébastien Canard, Matthieu Gaud, and Jacques Traoré, *Defeating malicious servers in a blind signatures based voting system*, FC 2006: 10th International Conference on Financial Cryptography and Data Security (Anguilla, British West Indies) (Giovanni Di Crescenzo and Avi Rubin, eds.), Lecture Notes in Computer Science, vol. 4107, Springer, Berlin, Germany, February 27 – March 2, 2006, pp. 148–153. (Cited on page 39, 40, 178, 179.)

[CH07] Ran Canetti and Susan Hohenberger, *Chosen-ciphertext secure proxy re-encryption*, ACM CCS 07: 14th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, eds.), ACM Press, October 28–31, 2007, pp. 185–194. (Cited on page 23, 69, 70, 71, 72, 74, 76, 85, 87, 92.)

[Cha82] David Chaum, *Blind signatures for untraceable payments*, Advances in Cryptology – CRYPTO'82 (Santa Barbara, CA, USA) (David Chaum, Ronald L. Rivest, and Alan T. Sherman, eds.), Plenum Press, New York, USA, 1982, pp. 199–203. (Cited on page 10, 18, 149, 177, 214, 306.)

[Cha83] ———, *Blind signature system*, Advances in Cryptology – CRYPTO'83 (Santa Barbara, CA, USA) (David Chaum, ed.), Plenum Press, New York, USA, 1983, p. 153. (Cited on page 10.)

[Cha85] ———, *Security without identification: Transaction systems to make big brother obsolete*, Commun. ACM **28** (1985), no. 10, 1030–1044. (Cited on page 35.)

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz, *Chosen-ciphertext security from identity-based encryption*, Advances in Cryptology – EUROCRYPT 2004 (Interlaken, Switzerland) (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science, vol. 3027, Springer, Berlin, Germany, May 2–6, 2004, pp. 207–222. (Cited on page 75, 76, 111, 119, 120, 121, 128.)

[CHK05a] ———, *Adaptively-secure, non-interactive public-key encryption*, TCC 2005: 2nd Theory of Cryptography Conference (Cambridge, MA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 3378, Springer, Berlin, Germany, February 10–12, 2005, pp. 150–168. (Cited on page 108.)

[CHK+05b] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie, *Universally composable password-based key exchange*, Advances in Cryptology – EUROCRYPT 2005 (Aarhus, Denmark) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, Berlin, Germany, May 22–26, 2005, pp. 404–421. (Cited on page 46, 50, 248, 249, 250, 255, 258, 260, 279, 280, 294, 305.)

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya, *Compact e-cash*, Advances in Cryptology – EUROCRYPT 2005 (Aarhus, Denmark) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, Berlin, Germany, May 22–26, 2005, pp. 302–321. (Cited on page 35.)

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik, *Secret handshakes from CA-oblivious encryption*, Advances in Cryptology – ASIACRYPT 2004 (Jeju Island, Korea) (Pil Joong Lee, ed.), Lecture Notes in Computer Science, vol. 3329, Springer, Berlin, Germany, December 5–9, 2004, pp. 293–307. (Cited on page 214.)

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen, *Relaxing chosen-ciphertext security*, Advances in Cryptology – CRYPTO 2003 (Santa Barbara, CA, USA) (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, Berlin, Germany, August 17–21, 2003, pp. 565–582. (Cited on page 17, 71, 72, 76, 114.)

[CKP07] Ronald Cramer, Eike Kiltz, and Carles Padró, *A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra*, Advances in Cryptology – CRYPTO 2007 (Santa Barbara, CA, USA) (Alfred Menezes, ed.), Lecture Notes in Computer Science, vol. 4622, Springer, Berlin, Germany, August 19–23, 2007, pp. 613–630. (Cited on page 251.)

[CKS08] David Cash, Eike Kiltz, and Victor Shoup, *The twin Diffie-Hellman problem and applications*, Advances in Cryptology – EUROCRYPT 2008 (Istanbul, Turkey) (Nigel P. Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer, Berlin, Germany, April 13–17, 2008, pp. 127–145. (Cited on page 128, 130.)

[CKS11] Jan Camenisch, Stephan Krenn, and Victor Shoup, *A framework for practical universally composable zero-knowledge protocols*, Advances in Cryptology – ASIACRYPT 2011 (Seoul, South Korea) (Dong Hoon Lee and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 7073, Springer, Berlin, Germany, December 4–8, 2011, pp. 449–467. (Cited on page 260.)

[CL01] Jan Camenisch and Anna Lysyanskaya, *An efficient system for non-transferable anonymous credentials with optional anonymity revocation*, Advances in Cryptology – EUROCRYPT 2001 (Innsbruck, Austria) (Birgit Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer, Berlin, Germany, May 6–10, 2001, pp. 93–118. (Cited on page 35, 36.)

[CL02a] ———, *Dynamic accumulators and application to efficient revocation of anonymous credentials*, Advances in Cryptology – CRYPTO 2002 (Santa Barbara, CA, USA) (Moti Yung, ed.), Lecture Notes in Computer Science, vol. 2442, Springer, Berlin, Germany, August 18–22, 2002, pp. 61–76. (Cited on page 35.)

[CL02b] ———, *A signature scheme with efficient protocols*, SCN 02: 3rd International Conference on Security in Communication Networks (Amalfi, Italy) (Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, eds.), Lecture Notes in Computer Science, vol. 2576, Springer, Berlin, Germany, September 12–13, 2002, pp. 268–289. (Cited on page 35.)

[CL04] ———, *Signature schemes and anonymous credentials from bilinear maps*, Advances in Cryptology – CRYPTO 2004 (Santa Barbara, CA, USA) (Matthew Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer, Berlin, Germany, August 15–19, 2004, pp. 56–72. (Cited on page 35, 36.)

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai, *Universally composable two-party and multi-party secure computation*, 34th Annual ACM Symposium on Theory of Computing (Montréal, Québec, Canada), ACM Press, May 19–21, 2002, pp. 494–503. (Cited on page 45, 46.)

[CMFP+10] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré, *On some incompatible properties of voting schemes*, Towards Trustworthy Elections, New Directions in Electronic Voting (David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, eds.), Lecture Notes in Computer Science, vol. 6000, Springer, 2010, pp. 191–199. (Cited on page 151.)

[Coc01] Clifford Cocks, *An identity based encryption scheme based on quadratic residues*, 8th IMA International Conference on Cryptography and Coding (Cirencester, UK) (Bahram Honary, ed.), Lecture Notes in Computer Science, vol. 2260, Springer, Berlin, Germany, December 17–19, 2001, pp. 360–363. (Cited on page 48.)

[Cor00] Jean-Sébastien Coron, *On the exact security of full domain hash*, Advances in Cryptology – CRYPTO 2000 (Santa Barbara, CA, USA) (Mihir Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer, Berlin, Germany, August 20–24, 2000, pp. 229–235. (Cited on page 97, 98, 103.)

[CP92] David Chaum and Torben P. Pedersen, *Transferred cash grows in size*, Advances in Cryptology – EUROCRYPT'92 (Balatonfüred, Hungary) (Rainer A. Rueppel, ed.), Lecture Notes in Computer Science, vol. 658, Springer, Berlin, Germany, May 24–28, 1992, pp. 390–407. (Cited on page 35.)

[CPY06] Seung Geol Choi, Kunsoo Park, and Moti Yung, *Short traceable signatures based on bilinear pairings*, IWSEC 06: 1st International Workshop on Security, Advances in Information and Computer Security (Kyoto, Japan) (Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shin ichi Kawamura, eds.), Lecture Notes in Computer Science, vol. 4266, Springer, Berlin, Germany, October 23–24, 2006, pp. 88–103. (Cited on page 197.)

[CR03] Ran Canetti and Tal Rabin, *Universal composition with joint state*, Advances in Cryptology – CRYPTO 2003 (Santa Barbara, CA, USA) (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, Berlin, Germany, August 17–21, 2003, pp. 265–281. (Cited on page 250.)

[CS98] Ronald Cramer and Victor Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Advances in Cryptology – CRYPTO'98 (Santa Barbara, CA, USA) (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer, Berlin, Germany, August 23–27, 1998, pp. 13–25. (Cited on page 41, 44, 47, 118, 119, 128, 266, 273, 294, 311.)

[CS02] ———, *Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption*, Advances in Cryptology – EUROCRYPT 2002 (Amsterdam, The Netherlands) (Lars R. Knudsen, ed.), Lecture Notes in Computer Science, vol. 2332, Springer, Berlin, Germany, April 28 – May 2, 2002, pp. 45–64. (Cited on page 17, 41, 43, 109, 110, 111, 118, 119, 128, 139, 140, 213, 215, 223, 249, 251, 294, 296.)

[CS03] Jan Camenisch and Victor Shoup, *Practical verifiable encryption and decryption of discrete logarithms*, Advances in Cryptology – CRYPTO 2003 (Santa Barbara, CA, USA) (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, Berlin, Germany, August 17–21, 2003, pp. 126–144. (Cited on page 47.)

[CS06] Sanjit Chatterjee and Palash Sarkar, *Generalization of the selective-ID security model for HIBE protocols*, PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography (New York, NY, USA) (Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, eds.), Lecture Notes in Computer Science, vol. 3958, Springer, Berlin, Germany, April 24–26, 2006, pp. 241–256. (Cited on page 126, 128.)

[Cv91] David Chaum and Eugène van Heyst, *Group signatures*, Advances in Cryptology – EUROCRYPT'91 (Brighton, UK) (Donald W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer, Berlin, Germany, April 8–11, 1991, pp. 257–265. (Cited on page 32, 195.)

[Dam91] Ivan Damgård, *Towards practical public key systems secure against chosen ciphertext attacks*, Advances in Cryptology – CRYPTO'91 (Santa Barbara, CA, USA) (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, Berlin, Germany, August 11–15, 1991, pp. 445–456. (Cited on page 94, 182.)

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor, *Non-malleable cryptography (extended abstract)*, 23rd Annual ACM Symposium on Theory of Computing (New Orleans, Louisiana, USA), ACM Press, May 6–8, 1991, pp. 542–552. (Cited on page 15, 118, 142, 143.)

[DDN00] ———, *Nonmalleable cryptography*, SIAM J. Comput. **30** (2000), no. 2, 391–437. (Cited on page 15.)

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai, *Robust non-interactive zero knowledge*, Advances in Cryptology – CRYPTO 2001 (Santa Barbara, CA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer, Berlin, Germany, August 19–23, 2001, pp. 566–598. (Cited on page 142.)

[Den06] Alexander W. Dent, *The hardness of the dhk problem in the generic group model*, Cryptology ePrint Archive, Report 2006/156, 2006, `http://eprint.iacr.org/`. (Cited on page 94.)

[DH76] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), no. 6, 644–654. (Cited on page 6, 49, 294.)

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky, *Non-interactive and non-malleable commitment*, 30th Annual ACM Symposium on Theory of Computing (Dallas, Texas, USA), ACM Press, May 23–26, 1998, pp. 141–150. (Cited on page 15, 141.)

[DJ01] Ivan Damgård and Mats Jurik, *A generalisation, a simplification and some applications of Paillier's probabilistic public-key system*, PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography (Cheju Island, South Korea) (Kwangjo Kim, ed.), Lecture Notes in Computer Science, vol. 1992, Springer, Berlin, Germany, February 13–15, 2001, pp. 119–136. (Cited on page 116, 126, 127, 146.)

[DK10] Stéphanie Delaune and Steve Kremer, *Formalising security properties in electronic voting protocols*, Deliverable AVOTE 1.2, (ANR-07-SESU-002), April 2010, 17 pages, 2010. (Cited on page 151.)

[DKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith, *Efficient and non-interactive non-malleable commitment*, Advances in Cryptology – EUROCRYPT 2001 (Innsbruck, Austria) (Birgit Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer, Berlin, Germany, May 6–10, 2001, pp. 40–59. (Cited on page 15.)

[DM95] Burgess Davis and David McDonald, *An elementary proof of the local central limit theorem.*, J. Theor. Probab. **8** (1995), no. 3, 693–701. (Cited on page 171.)

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky, *Single database private information retrieval implies oblivious transfer*, Advances in Cryptology – EUROCRYPT 2000 (Bruges, Belgium) (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer, Berlin, Germany, May 14–18, 2000, pp. 122–138. (Cited on page 110, 117.)

[DN02] Ivan Damgård and Jesper Buus Nielsen, *Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor*, Advances in Cryptology – CRYPTO 2002 (Santa Barbara, CA, USA) (Moti Yung, ed.), Lecture Notes in Computer Science, vol. 2442, Springer, Berlin, Germany, August 18–22, 2002, pp. 581–596. (Cited on page 39, 45, 47, 150, 153.)

[DN07] Cynthia Dwork and Moni Naor, *Zaps and their applications*, SIAM J. Comput. **36** (2007), no. 6, 1513–1543. (Cited on page 227.)

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer, *Magic functions: In memoriam: Bernard m. dwork 1923–1998*, Journal of the ACM **50** (2003), no. 6, 852–921. (Cited on page 108.)

[DP06] Cécile Delerablée and David Pointcheval, *Dynamic fully anonymous short group signatures*, Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam (Hanoi, Vietnam) (Phong Q. Nguyen, ed.), Lecture Notes in Computer Science, vol. 4341, Springer, Berlin, Germany, September 25–28, 2006, pp. 193–210. (Cited on page 32, 196.)

[DPR+13] Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs, *Security analysis of pseudo-random number generators with input: /dev/random is not robust*, ACM CCS 13: 20th Conference on Computer and Communications Security (Berlin, Germany) (Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, eds.), ACM Press, November 4–8, 2013, pp. 647–658. (Cited on page 3.)

[Duc10] Léo Ducas, *Anonymity from asymmetry: New constructions for anonymous HIBE*, Topics in Cryptology – CT-RSA 2010 (San Francisco, CA, USA) (Josef Pieprzyk, ed.), Lecture Notes in Computer Science, vol. 5985, Springer, Berlin, Germany, March 1–5, 2010, pp. 148–164. (Cited on page 261.)

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy, *A verifiable random function with short proofs and keys*, PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography (Les Diablerets, Switzerland) (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 3386, Springer, Berlin, Germany, January 23–26, 2005, pp. 416–431. (Cited on page 75.)

[EG14] Alex Escala and Jens Groth, *Fine-tuning Groth-Sahai proofs*, PKC 2014: 17th International Workshop on Theory and Practice in Public Key Cryptography, Lecture Notes in Computer Science, Springer, Berlin, Germany, 2014, pp. 630–649. (Cited on page 55.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar, *An algebraic framework for Diffie-Hellman assumptions*, Advances in Cryptology – CRYPTO 2013, Part II (Santa Barbara, CA, USA) (Ran Canetti and Juan A. Garay, eds.), Lecture Notes in Computer Science, vol. 8043, Springer, Berlin, Germany, August 18–22, 2013, pp. 129–147. (Cited on page 55.)

[EO94] Tony Eng and Tatsuaki Okamoto, *Single-term divisible electronic coins*, Advances in Cryptology – EUROCRYPT'94 (Perugia, Italy) (Alfredo De Santis, ed.), Lecture Notes in Computer Science, vol. 950, Springer, Berlin, Germany, May 9–12, 1994, pp. 306–319. (Cited on page 35.)

[FF00] Marc Fischlin and Roger Fischlin, *Efficient non-malleable commitment schemes*, Advances in Cryptology – CRYPTO 2000 (Santa Barbara, CA, USA) (Mihir Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer, Berlin, Germany, August 20–24, 2000, pp. 413–431. (Cited on page 15.)

[FGHP09] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen, *Practical short signature batch verification*, Topics in Cryptology – CT-RSA 2009 (San Francisco, CA, USA) (Marc Fischlin, ed.), Lecture Notes in Computer Science, vol. 5473, Springer, Berlin, Germany, April 20–24, 2009, pp. 309–324. (Cited on page 31.)

[FHKW10] Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee, *Encryption schemes secure against chosen-ciphertext selective opening attacks*, Advances in Cryptology – EUROCRYPT 2010 (French Riviera) (Henri Gilbert, ed.), Lecture Notes in Computer Science, vol. 6110, Springer, Berlin, Germany, May 30 – June 3, 2010, pp. 381–402. (Cited on page 109, 111.)

[FI05] Jun Furukawa and Hideki Imai, *An efficient group signature scheme from bilinear maps*, ACISP 05: 10th Australasian Conference on Information Security and Privacy (Brisbane, Queensland, Australia) (Colin Boyd and Juan Manuel González Nieto, eds.), Lecture Notes in Computer Science, vol. 3574, Springer, Berlin, Germany, July 4–6, 2005, pp. 455–467. (Cited on page 32, 196.)

[Fia89] Amos Fiat, *Batch RSA*, Advances in Cryptology – CRYPTO'89 (Santa Barbara, CA, USA) (Gilles Brassard, ed.), Lecture Notes in Computer Science, vol. 435, Springer, Berlin, Germany, August 20–24, 1989, pp. 175–185. (Cited on page 31.)

[Fis05] Marc Fischlin, *Completely non-malleable schemes*, ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming (Lisbon, Portugal) (Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 3580, Springer, Berlin, Germany, July 11–15, 2005, pp. 779–790. (Cited on page 15.)

[Fis06] ———, *Round-optimal composable blind signatures in the common reference string model*, Advances in Cryptology – CRYPTO 2006 (Santa Barbara, CA, USA) (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer, Berlin, Germany, August 20–24, 2006, pp. 60–77. (Cited on page 37, 150, 214.)

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis, *Non-interactive and re-usable universally composable string commitments with adaptive security*, Advances in Cryptology – ASIACRYPT 2011 (Seoul, South Korea) (Dong Hoon Lee and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 7073, Springer, Berlin, Germany, December 4–8, 2011, pp. 468–485. (Cited on page 47.)

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia, *Robust encryption, revisited*, PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography (Nara, Japan) (Kaoru Kurosawa and Goichiro Hanaoka, eds.), Lecture Notes in Computer Science, vol. 7778, Springer, Berlin, Germany, February 26 – March 1, 2013, pp. 352–368. (Cited on page 18.)

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto, *Statistical zero knowledge protocols to prove modular polynomial relations*, Advances in Cryptology – CRYPTO'97 (Santa Barbara, CA, USA) (Burton S. Kaliski Jr., ed.), Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, Germany, August 17–21, 1997, pp. 16–30. (Cited on page 35.)

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud, *Transferable constant-size fair e-cash*, CANS 09: 8th International Conference on Cryptology and Network Security (Kanazawa, Japan) (Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, eds.), Lecture Notes in Computer Science, vol. 5888, Springer, Berlin, Germany, December 12–14, 2009, pp. 226–247. (Cited on page 27, 32, 33, 35, 182.)

[FS86] Amos Fiat and Adi Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology – CRYPTO'86 (Santa Barbara, CA, USA) (Andrew M. Odlyzko, ed.), Lecture Notes in Computer Science, vol. 263, Springer, Berlin, Germany, August 1986, pp. 186–194. (Cited on page 34, 35, 197.)

[Fuc09] Georg Fuchsbauer, *Automorphic signatures in bilinear groups and an application to round-optimal blind signatures*, Cryptology ePrint Archive, Report 2009/320, 2009, `http://eprint.iacr.org/2009/320`. (Cited on page 37, 40, 177, 178, 182, 183, 186, 188.)

[Fuc11] ———, *Commuting signatures and verifiable encryption*, Advances in Cryptology – EUROCRYPT 2011 (Tallinn, Estonia) (Kenneth G. Paterson, ed.), Lecture Notes in Computer Science, vol. 6632, Springer, Berlin, Germany, May 15–19, 2011, pp. 224–245. (Cited on page 21, 36, 39, 150.)

[FV10] Georg Fuchsbauer and Damien Vergnaud, *Fair blind signatures without random oracles*, AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa (Stellenbosch, South Africa) (Daniel J. Bernstein and Tanja Lange, eds.), Lecture Notes in Computer Science, vol. 6055, Springer, Berlin, Germany, May 3–6, 2010, pp. 16–33. (Cited on page 27, 39, 177.)

[FVZ13] Pierre-Alain Fouque, Damien Vergnaud, and Jean-Christophe Zapalowicz, *Time/memory/data tradeoffs for variants of the rsa problem*, Computing and Combinatorics, 19th International Conference, COCOON 2013 (Ding-Zhu Du and Guochuan Zhang, eds.), Lecture Notes in Computer Science, vol. 7936, Springer, 2013, pp. 651–662. (Cited on page 3.)

[GA07] Matthew Green and Giuseppe Ateniese, *Identity-based proxy re-encryption*, ACNS 07: 5th International Conference on Applied Cryptography and Network Security (Zhuhai, China) (Jonathan Katz and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4521, Springer, Berlin, Germany, June 5–8, 2007, pp. 288–306. (Cited on page 71, 87.)

[Gam85] Taher El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory **31** (1985), no. 4, 469–472. (Cited on page 7, 15, 22, 23, 48, 70.)

[Gen09] Craig Gentry, *Fully homomorphic encryption using ideal lattices*, 41st Annual ACM Symposium on Theory of Computing (Bethesda, Maryland, USA) (Michael Mitzenmacher, ed.), ACM Press, May 31 – June 2, 2009, pp. 169–178. (Cited on page 15, 55.)

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi, *Candidate multilinear maps from ideal lattices*, Advances in Cryptology – EUROCRYPT 2013 (Athens, Greece) (Thomas Johansson and Phong Q. Nguyen, eds.), Lecture Notes in Computer Science, vol. 7881, Springer, Berlin, Germany, May 26–30, 2013, pp. 1–17. (Cited on page 55, 300, 318, 319, 320.)

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters, *Candidate indistinguishability obfuscation and functional encryption for all circuits*, 54th Annual Symposium on Foundations of Computer Science (Berkeley, CA, USA), IEEE Computer Society Press, October 26–29, 2013, pp. 40–49. (Cited on page 55.)

[GJJS04] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson, *Universal re-encryption for mixnets*, Topics in Cryptology – CT-RSA 2004 (San Francisco, CA, USA) (Tatsuaki Okamoto, ed.), Lecture Notes in Computer Science, vol. 2964, Springer, Berlin, Germany, February 23–27, 2004, pp. 163–178. (Cited on page 17, 18, 114.)

[GK03] Shafi Goldwasser and Yael Tauman Kalai, *On the (in)security of the Fiat-Shamir paradigm*, 44th Annual Symposium on Foundations of Computer Science (Cambridge, Massachusetts, USA), IEEE Computer Society Press, October 11–14, 2003, pp. 102–115. (Cited on page 36, 197.)

[GL03] Rosario Gennaro and Yehuda Lindell, *A framework for password-based authenticated key exchange*, Advances in Cryptology – EUROCRYPT 2003 (Warsaw, Poland) (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer, Berlin, Germany, May 4–8, 2003, `http://eprint.iacr.org/2003/032.ps.gz`, pp. 524–543. (Cited on page 41, 43, 44, 46, 50, 214, 215, 229, 248, 251, 263, 294, 296, 299.)

[GL06] ———, *A framework for password-based authenticated key exchange*, ACM Transactions on Information and System Security **9** (2006), no. 2, 181–234. (Cited on page 48, 214.)

[GL07] Jens Groth and Steve Lu, *A non-interactive shuffle with pairing based verifiability*, Advances in Cryptology – ASIACRYPT 2007 (Kuching, Malaysia) (Kaoru Kurosawa, ed.), Lecture Notes in Computer Science, vol. 4833, Springer, Berlin, Germany, December 2–6, 2007, pp. 51–67. (Cited on page 40, 179.)

[GM84] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption*, J. Comput. Syst. Sci. **28** (1984), no. 2, 270–299. (Cited on page 3, 6, 15, 153, 215.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM J. Comput. **17** (1988), no. 2, 281–308. (Cited on page 8, 9, 154, 160, 215, 266.)

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang, *Strengthening zero-knowledge protocols using signatures*, Journal of Cryptology **19** (2006), no. 2, 169–209. (Cited on page 260.)

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai, *Non-interactive zaps and new techniques for NIZK*, Advances in Cryptology – CRYPTO 2006 (Santa Barbara, CA, USA) (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer, Berlin, Germany, August 20–24, 2006, pp. 97–111. (Cited on page 16, 142, 227.)

[GOS06b] ———, *Perfect non-interactive zero knowledge for NP*, Advances in Cryptology – EUROCRYPT 2006 (St. Petersburg, Russia) (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 4004, Springer, Berlin, Germany, May 28 – June 1, 2006, pp. 339–358. (Cited on page 32.)

[Goy07] Vipul Goyal, *Reducing trust in the PKG in identity based cryptosystems*, Advances in Cryptology – CRYPTO 2007 (Santa Barbara, CA, USA) (Alfred Menezes, ed.), Lecture Notes in Computer Science, vol. 4622, Springer, Berlin, Germany, August 19–23, 2007, pp. 430–447. (Cited on page 24.)

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart, *Pairings for cryptographers*, Discrete Applied Mathematics **156** (2008), no. 16, 3113–3121. (Cited on page 5, 28, 31, 49, 215, 225.)

[Gro04] Jens Groth, *Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems*, TCC 2004: 1st Theory of Cryptography Conference (Cambridge, MA, USA) (Moni Naor, ed.), Lecture Notes in Computer Science, vol. 2951, Springer, Berlin, Germany, February 19–21, 2004, pp. 152–170. (Cited on page 17, 114.)

[Gro06] ———, *Simulation-sound NIZK proofs for a practical language and constant size group signatures*, Advances in Cryptology – ASIACRYPT 2006 (Shanghai, China) (Xuejia Lai and Kefei Chen, eds.), Lecture Notes in Computer Science, vol. 4284, Springer, Berlin, Germany, December 3–7, 2006, pp. 444–459. (Cited on page 32, 33, 186, 191, 196, 197.)

[Gro07] ———, *Fully anonymous group signatures without random oracles*, Advances in Cryptology – ASIACRYPT 2007 (Kuching, Malaysia) (Kaoru Kurosawa, ed.), Lecture Notes in Computer Science, vol. 4833, Springer, Berlin, Germany, December 2–6, 2007, pp. 164–180. (Cited on page 32, 33, 40, 178, 184, 196, 197, 213.)

[Gro09] ———, *Homomorphic trapdoor commitments to group elements*, Cryptology ePrint Archive, Report 2009/007, 2009, `http://eprint.iacr.org/2009/007`. (Cited on page 186, 191.)

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh, *Round optimal blind signatures*, Advances in Cryptology – CRYPTO 2011 (Santa Barbara, CA, USA) (Phillip Rogaway, ed.), Lecture Notes in Computer Science, vol. 6841, Springer, Berlin, Germany, August 14–18, 2011, pp. 630–648. (Cited on page 150, 227.)

[GS06] R Granger and N.P. Smart, *On computing products of pairings*, Cryptology ePrint Archive, Report 2006/172, 2006, `http://eprint.iacr.org/`. (Cited on page 82, 96.)

[GS08] Jens Groth and Amit Sahai, *Efficient non-interactive proof systems for bilinear groups*, Advances in Cryptology – EUROCRYPT 2008 (Istanbul, Turkey) (Nigel P. Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer, Berlin, Germany, April 13–17, 2008, pp. 415–432. (Cited on page 27, 28, 36, 39, 47, 51, 150, 151, 158, 162, 163, 177, 178, 183, 197, 200, 201, 202, 207, 213, 253, 295, 299, 309.)

[GS12] ———, *Efficient noninteractive proof systems for bilinear groups*, SIAM J. Comput. **41** (2012), no. 5, 1193–1232. (Cited on page 213.)

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi, *Groth-Sahai proofs revisited*, PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography (Paris, France) (Phong Q. Nguyen and David Pointcheval, eds.), Lecture Notes in Computer Science, vol. 6056, Springer, Berlin, Germany, May 26–28, 2010, pp. 177–192. (Cited on page 55.)

[GT03] Matthieu Gaud and Jacques Traoré, *On the anonymity of fair offline e-cash systems*, FC 2003: 7th International Conference on Financial Cryptography (Guadeloupe, French West Indies) (Rebecca Wright, ed.), Lecture Notes in Computer Science, vol. 2742, Springer, Berlin, Germany, January 27–30, 2003, pp. 34–50. (Cited on page 39, 40, 178, 179.)

[GV12] Aurore Guillevic and Damien Vergnaud, *Genus 2 hyperelliptic curve families with explicit jacobian order evaluation and pairing-friendly constructions*, PAIRING 2012: 5th International Conference on Pairing-based Cryptography (Cologne, Germany) (Michel Abdalla and Tanja Lange, eds.), Lecture Notes in Computer Science, vol. 7708, Springer, Berlin, Germany, May 16–18, 2012, pp. 234–253. (Cited on page 3.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby, *A pseudorandom generator from any one-way function*, SIAM J. Comput. **28** (1999), no. 4, 1364–1396. (Cited on page 218.)

[HK08] Dennis Hofheinz and Eike Kiltz, *Programmable hash functions and their applications*, Advances in Cryptology – CRYPTO 2008 (Santa Barbara, CA, USA) (David Wagner, ed.), Lecture Notes in Computer Science, vol. 5157, Springer, Berlin, Germany, August 17–21, 2008, pp. 21–38. (Cited on page 34, 39, 151, 166, 168, 169, 171, 197, 240, 318.)

[HK12] Shai Halevi and Yael Tauman Kalai, *Smooth projective hashing and two-message oblivious transfer*, Journal of Cryptology **25** (2012), no. 1, 158–193. (Cited on page 110, 118.)

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell, *Concurrently-secure blind signatures without random oracles or setup assumptions*, TCC 2007: 4th Theory of Cryptography Conference (Amsterdam, The Netherlands) (Salil P. Vadhan, ed.), Lecture Notes in Computer Science, vol. 4392, Springer, Berlin, Germany, February 21–24, 2007, pp. 323–341. (Cited on page 236, 307.)

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud, *Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security*, Advances in Cryptology – ASIACRYPT 2011 (Seoul, South Korea) (Dong Hoon Lee and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 7073, Springer, Berlin, Germany, December 4–8, 2011, pp. 70–88. (Cited on page 15, 17, 107.)

[Hof11] Dennis Hofheinz, *Possibility and impossibility results for selective decommitments*, Journal of Cryptology **24** (2011), no. 3, 470–516. (Cited on page 108, 116.)

[Hof12] ———, *All-but-many lossy trapdoor functions*, Advances in Cryptology – EUROCRYPT 2012 (Cambridge, UK) (David Pointcheval and Thomas Johansson, eds.), Lecture Notes in Computer Science, vol. 7237, Springer, Berlin, Germany, April 15–19, 2012, pp. 209–227. (Cited on page 111, 124, 131.)

[Hoh06] Susan Hohenberger, *Advances in signatures, encryption, and e-cash from bilinear groups*, Ph.D. Thesis, MIT, 2006. (Cited on page 23, 24, 71, 87.)

[HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan, *Securely obfuscating re-encryption*, TCC 2007: 4th Theory of Cryptography Conference (Amsterdam, The Netherlands) (Salil P. Vadhan, ed.), Lecture Notes in Computer Science, vol. 4392, Springer, Berlin, Germany, February 21–24, 2007, pp. 233–252. (Cited on page 70, 87.)

[HT07] Emeline Hufschmitt and Jacques Traoré, *Fair blind signatures revisited*, PAIRING 2007: 1st International Conference on Pairing-based Cryptography (Tokyo, Japan) (Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, eds.), Lecture Notes in Computer Science, vol. 4575, Springer, Berlin, Germany, July 2–4, 2007, pp. 268–292. (Cited on page 39, 177, 178, 180, 181, 182.)

[ID03] Anca Ivan and Yevgeniy Dodis, *Proxy cryptography revisited*, ISOC Network and Distributed System Security Symposium – NDSS 2003 (San Diego, California, USA), The Internet Society, February 5–7, 2003. (Cited on page 21, 22, 25, 70, 88.)

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky, *Sufficient conditions for collision-resistant hashing*, TCC 2005: 2nd Theory of Cryptography Conference (Cambridge, MA, USA) (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 3378, Springer, Berlin, Germany, February 10–12, 2005, pp. 445–456. (Cited on page 118.)

[ILV11] Malika Izabachène, Benoît Libert, and Damien Vergnaud, *Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes*, 13th IMA International Conference on Cryptography and Coding (Oxford, UK) (Liqun Chen, ed.), Lecture Notes in Computer Science, vol. 7089, Springer, Berlin, Germany, December 12–15, 2011, pp. 431–450. (Cited on page 36, 53, 54.)

[IPV10] Malika Izabachène, David Pointcheval, and Damien Vergnaud, *Mediated traceable anonymous encryption*, Progress in Cryptology - LATINCRYPT 2010: 1st International Conference on Cryptology

and Information Security in Latin America (Puebla, Mexico) (Michel Abdalla and Paulo S. L. M. Barreto, eds.), Lecture Notes in Computer Science, vol. 6212, Springer, Berlin, Germany, August 8–11, 2010, pp. 40–60. (Cited on page 15, 18.)

[Jak99] Markus Jakobsson, *On quorum controlled asymmetric proxy re-encryption*, PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography (Kamakura, Japan) (Hideki Imai and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1560, Springer, Berlin, Germany, March 1–3, 1999, pp. 112–121. (Cited on page 21, 70.)

[JG02] Ari Juels and Jorge Guajardo, *RSA key generation with verifiable randomness*, PKC 2002: 5th International Workshop on Theory and Practice in Public Key Cryptography (Paris, France) (David Naccache and Pascal Paillier, eds.), Lecture Notes in Computer Science, vol. 2274, Springer, Berlin, Germany, February 12–14, 2002, pp. 357–374. (Cited on page 218.)

[JL09] Stanislaw Jarecki and Xiaomin Liu, *Private mutual authentication and conditional oblivious transfer*, Advances in Cryptology – CRYPTO 2009 (Santa Barbara, CA, USA) (Shai Halevi, ed.), Lecture Notes in Computer Science, vol. 5677, Springer, Berlin, Germany, August 16–20, 2009, pp. 90–107. (Cited on page 51, 248, 261.)

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky, *Security of blind digital signatures (extended abstract)*, Advances in Cryptology – CRYPTO'97 (Santa Barbara, CA, USA) (Burton S. Kaliski Jr., ed.), Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, Germany, August 17–21, 1997, pp. 150–164. (Cited on page 11, 237, 308.)

[Jou13] Antoine Joux, *Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields*, Advances in Cryptology – EUROCRYPT 2013 (Athens, Greece) (Thomas Johansson and Phong Q. Nguyen, eds.), Lecture Notes in Computer Science, vol. 7881, Springer, Berlin, Germany, May 26–30, 2013, pp. 177–193. (Cited on page 5.)

[JP13] Antoine Joux and Cécile Pierrot, *The special number field sieve in $\mathbb{F}_{p^n}$ - application to pairing-friendly constructions*, PAIRING 2013: 6th International Conference on Pairing-based Cryptography (Beijing, China) (Zhenfu Cao and Fangguo Zhang, eds.), Lecture Notes in Computer Science, vol. 8365, Springer, Berlin, Germany, November 22–24, 2013, pp. 45–61. (Cited on page 5.)

[JR12] Charanjit S. Jutla and Arnab Roy, *Relatively-sound NIZKs and password-based key-exchange*, PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography (Darmstadt, Germany) (Marc Fischlin, Johannes Buchmann, and Mark Manulis, eds.), Lecture Notes in Computer Science, vol. 7293, Springer, Berlin, Germany, May 21–23, 2012, pp. 485–503. (Cited on page 52, 295, 299.)

[JTV10] Marc Joye, Mehdi Tibouchi, and Damien Vergnaud, *Huff's model for elliptic curves*, ANTS (Guillaume Hanrot, François Morain, and Emmanuel Thomé, eds.), Lecture Notes in Computer Science, vol. 6197, Springer, 2010, pp. 234–250. (Cited on page 3.)

[Kal05] Yael Tauman Kalai, *Smooth projective hashing and two-message oblivious transfer*, Advances in Cryptology – EUROCRYPT 2005 (Aarhus, Denmark) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, Berlin, Germany, May 22–26, 2005, pp. 78–95. (Cited on page 41, 118, 214.)

[KG06] Eike Kiltz and David Galindo, *Direct chosen-ciphertext secure identity-based key encapsulation without random oracles*, ACISP 06: 11th Australasian Conference on Information Security and Privacy (Melbourne, Australia) (Lynn Margaret Batten and Reihaneh Safavi-Naini, eds.), Lecture Notes in Computer Science, vol. 4058, Springer, Berlin, Germany, July 3–5, 2006, pp. 336–347. (Cited on page 82.)

[Kil06] Eike Kiltz, *Chosen-ciphertext security from tag-based encryption*, TCC 2006: 3rd Theory of Cryptography Conference (New York, NY, USA) (Shai Halevi and Tal Rabin, eds.), Lecture Notes in Computer Science, vol. 3876, Springer, Berlin, Germany, March 4–7, 2006, pp. 581–600. (Cited on page 40, 76, 82, 120, 121, 178, 184.)

[KJP06] Sébastien Kunz-Jacques and David Pointcheval, *A new key exchange protocol based on MQV assuming public computations*, SCN 06: 5th International Conference on Security in Communication Networks (Maiori, Italy) (Roberto De Prisco and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4116, Springer, Berlin, Germany, September 6–8, 2006, pp. 186–200. (Cited on page 93, 94, 104.)

[KMPR05] Eike Kiltz, Anton Mityagin, Saurabh Panjwani, and Barath Raghavan, *Append-only signatures*, ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming (Lisbon, Portugal) (Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 3580, Springer, Berlin, Germany, July 11–15, 2005, pp. 434–445. (Cited on page 207.)

[KN08] Gillat Kol and Moni Naor, *Cryptography and game theory: Designing protocols for exchanging information*, TCC 2008: 5th Theory of Cryptography Conference (San Francisco, CA, USA) (Ran Canetti, ed.), Lecture Notes in Computer Science, vol. 4948, Springer, Berlin, Germany, March 19–21, 2008, pp. 320–339. (Cited on page 16, 113, 118, 140.)

[KO97] Eyal Kushilevitz and Rafail Ostrovsky, *Replication is NOT needed: SINGLE database, computationally-private information retrieval*, 38th Annual Symposium on Foundations of Computer Science (Miami Beach, Florida), IEEE Computer Society Press, October 19–22, 1997, pp. 364–373. (Cited on page 118.)

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung, *Efficient password-authenticated key exchange using human-memorable passwords*, Advances in Cryptology – EUROCRYPT 2001 (Innsbruck, Austria) (Birgit Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer, Berlin, Germany, May 6–10, 2001, pp. 475–494. (Cited on page 49, 294.)

[KR00] Hugo Krawczyk and Tal Rabin, *Chameleon signatures*, ISOC Network and Distributed System Security Symposium – NDSS 2000 (San Diego, California, USA), The Internet Society, February 2–4, 2000. (Cited on page 119, 121.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters, *Predicate encryption supporting disjunctions, polynomial equations, and inner products*, Advances in Cryptology – EUROCRYPT 2008 (Istanbul, Turkey) (Nigel P. Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer, Berlin, Germany, April 13–17, 2008, pp. 146–162. (Cited on page 36.)

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung, *Traceable signatures*, Advances in Cryptology – EUROCRYPT 2004 (Interlaken, Switzerland) (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science, vol. 3027, Springer, Berlin, Germany, May 2–6, 2004, pp. 571–589. (Cited on page 197, 210.)

[KTY07] ———, *Group encryption*, Advances in Cryptology – ASIACRYPT 2007 (Kuching, Malaysia) (Kaoru Kurosawa, ed.), Lecture Notes in Computer Science, vol. 4833, Springer, Berlin, Germany, December 2–6, 2007, pp. 181–199. (Cited on page 17, 18.)

[KV11] Jonathan Katz and Vinod Vaikuntanathan, *Round-optimal password-based authenticated key exchange*, TCC 2011: 8th Theory of Cryptography Conference (Providence, RI, USA) (Yuval Ishai, ed.), Lecture Notes in Computer Science, vol. 6597, Springer, Berlin, Germany, March 28–30, 2011, pp. 293–310. (Cited on page 43, 51, 52, 294, 296, 299, 305, 306, 312, 313.)

[KY04] Aggelos Kiayias and Moti Yung, *Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders*, Cryptology ePrint Archive, Report 2004/076, 2004, `http://eprint.iacr.org/2004/076`. (Cited on page 32, 196.)

[KZ09] Aggelos Kiayias and Hong-Sheng Zhou, *Zero-knowledge proofs with witness elimination*, PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography (Irvine, CA, USA) (Stanislaw Jarecki and Gene Tsudik, eds.), Lecture Notes in Computer Science, vol. 5443, Springer, Berlin, Germany, March 18–20, 2009, pp. 124–138. (Cited on page 53.)

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh, *Oblivious signature-based envelope*, 22nd ACM Symposium Annual on Principles of Distributed Computing (Boston, Massachusetts, USA) (Elizabeth Borowsky and Sergio Rajsbaum, eds.), ACM Press, July 13–16, 2003, pp. 182–189. (Cited on page 48, 214, 215, 216, 218, 225.)

[Lin06] Yehuda Lindell, *A simpler construction of CCA2-secure public-key encryption under general assumptions*, Journal of Cryptology **19** (2006), no. 3, 359–377. (Cited on page 142.)

[Lin11a] ———, *Highly-efficient universally-composable commitments based on the DDH assumption*, Advances in Cryptology – EUROCRYPT 2011 (Tallinn, Estonia) (Kenneth G. Paterson, ed.), Lecture Notes in Computer Science, vol. 6632, Springer, Berlin, Germany, May 15–19, 2011, pp. 446–466. (Cited on page 47, 249, 251, 252, 271, 272.)

[Lin11b] ———, *Highly-efficient universally-composable commitments based on the DDH assumption*, Cryptology ePrint Archive, Report 2011/180, 2011, `http://eprint.iacr.org/2011/180`. (Cited on page 47.)

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters, *Sequential aggregate signatures and multisignatures without random oracles*, Advances in Cryptology – EUROCRYPT 2006 (St. Petersburg, Russia) (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 4004, Springer, Berlin, Germany, May 28 – June 1, 2006, pp. 465–485. (Cited on page 99, 159.)

[LV08a] Benoît Libert and Damien Vergnaud, *Multi-use unidirectional proxy re-signatures*, ACM CCS 08: 15th Conference on Computer and Communications Security (Alexandria, Virginia, USA) (Peng Ning, Paul F. Syverson, and Somesh Jha, eds.), ACM Press, October 27–31, 2008, pp. 511–520. (Cited on page 15, 26, 87.)

[LV08b] ———, *Tracing malicious proxies in proxy re-encryption*, PAIRING 2008: 2nd International Conference on Pairing-based Cryptography (Egham, UK) (Steven D. Galbraith and Kenneth G. Paterson, eds.), Lecture Notes in Computer Science, vol. 5209, Springer, Berlin, Germany, September 1–3, 2008, pp. 332–353. (Cited on page 15, 24.)

[LV08c] ———, *Unidirectional chosen-ciphertext secure proxy re-encryption*, PKC 2008: 11th International Conference on Theory and Practice of Public Key Cryptography (Barcelona, Spain) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 4939, Springer, Berlin, Germany, March 9–12, 2008, pp. 360–379. (Cited on page 15, 23.)

[LV09] ———, *Group signatures with verifier-local revocation and backward unlinkability in the standard model*, CANS 09: 8th International Conference on Cryptology and Network Security (Kanazawa, Japan) (Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, eds.), Lecture Notes in Computer Science, vol. 5888, Springer, Berlin, Germany, December 12–14, 2009, pp. 498–517. (Cited on page 27, 34, 195.)

[LV11] Benoît Libert and Damien Vergnaud, *Unidirectional chosen-ciphertext secure proxy re-encryption*, IEEE Transactions on Information Theory **57** (2011), no. 3, 1786–1802. (Cited on page 15, 23, 69.)

[LY09] Benoît Libert and Moti Yung, *Efficient traceable signatures in the standard model*, PAIRING 2009: 3rd International Conference on Pairing-based Cryptography (Palo Alto, CA, USA) (Hovav Shacham and Brent Waters, eds.), Lecture Notes in Computer Science, vol. 5671, Springer, Berlin, Germany, August 12–14, 2009, pp. 187–205. (Cited on page 197, 210.)

[LY12] ———, *Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions*, TCC 2012: 9th Theory of Cryptography Conference (Taormina, Sicily, Italy) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 7194, Springer, Berlin, Germany, March 19–21, 2012, pp. 75–93. (Cited on page 52, 295.)

[Man98] Eran Mann, *Private access to distributed information*, Master's thesis, Technion - Israel Institute of Technology, 1998. (Cited on page 110, 118.)

[MO97] Masahiro Mambo and Eiji Okamoto, *Proxy cryptosystems: Delegation of the power to decrypt ciphertexts*, IEICE Transactions **80-A** (1997), no. 1, 54–63. (Cited on page 21, 70.)

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang, *Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract)*, TCC 2004: 1st Theory of Cryptography Conference (Cambridge, MA, USA) (Moni Naor, ed.), Lecture Notes in Computer Science, vol. 2951, Springer, Berlin, Germany, February 19–21, 2004, pp. 171–190. (Cited on page 120.)

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto, *Proxy signatures for delegating signing operation*, ACM CCS 96: 3rd Conference on Computer and Communications Security (New Delhi, India), ACM Press, March 14–15, 1996, pp. 48–57. (Cited on page 25, 87, 88.)

[Nac07] David Naccache, *Secure and practical identity-based encryption*, IET Information Security **1** (2007), no. 2, 59–64. (Cited on page 169, 170.)

[Nao03] Moni Naor, *On cryptographic assumptions and challenges (invited talk)*, Advances in Cryptology – CRYPTO 2003 (Santa Barbara, CA, USA) (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, Berlin, Germany, August 17–21, 2003, pp. 96–109. (Cited on page 32, 34, 94, 178, 196, 197, 200.)

[Nat94] National Institute of Standards and Technology, *Digital signature standard, nist fips pub 186*, U.S. Department of Commerce, 1994. (Cited on page 48.)

[NF05] Toru Nakanishi and Nobuo Funabiki, *Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps*, Advances in Cryptology – ASIACRYPT 2005 (Chennai, India) (Bimal K. Roy, ed.), Lecture Notes in Computer Science, vol. 3788, Springer, Berlin, Germany, December 4–8, 2005, pp. 533–548. (Cited on page 34, 196, 197, 198, 199, 202.)

[NF06] ———, *A short verifier-local revocation group signature scheme with backward unlinkability*, IWSEC 06: 1st International Workshop on Security, Advances in Information and Computer Security (Kyoto, Japan) (Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shin ichi Kawamura, eds.), Lecture Notes in Computer Science, vol. 4266, Springer, Berlin, Germany, October 23–24, 2006, pp. 17–32. (Cited on page 34, 197.)

[NHS99] Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama, *Unlinkable electronic coupon protocol with anonymity control*, ISW'99: 2nd International Workshop on Information Security (Kuala Lumpur, Malaysia) (Masahiro Mambo and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1729, Springer, Berlin, Germany, November 1999, pp. 37–46. (Cited on page 35.)

[NP01] Moni Naor and Benny Pinkas, *Efficient oblivious transfer protocols*, 12th Annual ACM-SIAM Symposium on Discrete Algorithms (Washington, DC, USA) (S. Rao Kosaraju, ed.), ACM-SIAM, January 7–9, 2001, pp. 448–457. (Cited on page 118, 128, 137.)

[NR93] Kaisa Nyberg and Rainer A. Rueppel, *A new signature scheme based on the DSA giving message recovery*, ACM CCS 93: 1st Conference on Computer and Communications Security (Fairfax, Virginia, USA) (V. Ashby, ed.), ACM Press, November 3–5, 1993, pp. 58–61. (Cited on page 48.)

[NSN04] Lan Nguyen and Reihaneh Safavi-Naini, *Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings*, Advances in Cryptology – ASIACRYPT 2004 (Jeju Island, Korea) (Pil Joong Lee, ed.), Lecture Notes in Computer Science, vol. 3329, Springer, Berlin, Germany, December 5–9, 2004, pp. 372–386. (Cited on page 32, 196, 197.)

[NT06] Samad Nasserian and Gene Tsudik, *Revisiting oblivious signature-based envelopes*, FC 2006: 10th International Conference on Financial Cryptography and Data Security (Anguilla, British West Indies) (Giovanni Di Crescenzo and Avi Rubin, eds.), Lecture Notes in Computer Science, vol. 4107, Springer, Berlin, Germany, February 27 – March 2, 2006, pp. 221–235. (Cited on page 48.)

[NY90] Moni Naor and Moti Yung, *Public-key cryptosystems provably secure against chosen ciphertext attacks*, 22nd Annual ACM Symposium on Theory of Computing (Baltimore, Maryland, USA), ACM Press, May 14–16, 1990, pp. 427–437. (Cited on page 7, 44, 51, 111, 118, 141, 142, 143, 145, 295.)

[Oka06] Tatsuaki Okamoto, *Efficient blind and partially blind signatures without random oracles*, TCC 2006: 3rd Theory of Cryptography Conference (New York, NY, USA) (Shai Halevi and Tal Rabin, eds.), Lecture Notes in Computer Science, vol. 3876, Springer, Berlin, Germany, March 4–7, 2006, pp. 80–99. (Cited on page 37, 39, 150, 178, 181.)

[OMA+99] Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto, *An improvement on a practical secret voting scheme*, ISW'99: 2nd International Workshop on Information Security (Kuala Lumpur, Malaysia) (Masahiro Mambo and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1729, Springer, Berlin, Germany, November 1999, pp. 225–234. (Cited on page 40, 179.)

[OO89] Tatsuaki Okamoto and Kazuo Ohta, *Disposable zero-knowledge authentications and their applications to untraceable electronic cash*, Advances in Cryptology – CRYPTO'89 (Santa Barbara, CA, USA) (Gilles Brassard, ed.), Lecture Notes in Computer Science, vol. 435, Springer, Berlin, Germany, August 20–24, 1989, pp. 481–496. (Cited on page 35.)

[OO91] ———, *Universal electronic cash*, Advances in Cryptology – CRYPTO'91 (Santa Barbara, CA, USA) (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, Berlin, Germany, August 11–15, 1991, pp. 324–337. (Cited on page 35.)

[Pai99] Pascal Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, Advances in Cryptology – EUROCRYPT'99 (Prague, Czech Republic) (Jacques Stern, ed.), Lecture Notes in Computer Science, vol. 1592, Springer, Berlin, Germany, May 2–6, 1999, pp. 223–238. (Cited on page 15, 17, 110, 116, 146.)

[Ped91] Torben P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Advances in Cryptology – CRYPTO'91 (Santa Barbara, CA, USA) (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, Berlin, Germany, August 11–15, 1991, pp. 129–140. (Cited on page 36, 272, 279.)

[Pen11] Kun Peng, *A general, flexible and efficient proof of inclusion and exclusion*, Topics in Cryptology – CT-RSA 2011 (San Francisco, CA, USA) (Aggelos Kiayias, ed.), Lecture Notes in Computer Science, vol. 6558, Springer, Berlin, Germany, February 14–18, 2011, pp. 33–48. (Cited on page 53.)

[Poi12] David Pointcheval, *Password-based authenticated key exchange (invited talk)*, PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography (Darmstadt, Germany) (Marc Fischlin, Johannes Buchmann, and Mark Manulis, eds.), Lecture Notes in Computer Science, vol. 7293, Springer, Berlin, Germany, May 21–23, 2012, pp. 390–397. (Cited on page 49, 294.)

[PR05] Rafael Pass and Alon Rosen, *New and improved constructions of non-malleable cryptographic protocols*, 37th Annual ACM Symposium on Theory of Computing (Baltimore, Maryland, USA) (Harold N. Gabow and Ronald Fagin, eds.), ACM Press, May 22–24, 2005, pp. 533–542. (Cited on page 15.)

[PR07] Manoj Prabhakaran and Mike Rosulek, *Rerandomizable RCCA encryption*, Advances in Cryptology – CRYPTO 2007 (Santa Barbara, CA, USA) (Alfred Menezes, ed.), Lecture Notes in Computer Science, vol. 4622, Springer, Berlin, Germany, August 19–23, 2007, pp. 517–534. (Cited on page 17, 114.)

[PS00] David Pointcheval and Jacques Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptology **13** (2000), no. 3, 361–396. (Cited on page 10, 149, 155.)

[PV05] Pascal Paillier and Damien Vergnaud, *Discrete-log-based signatures may not be equivalent to discrete log*, Advances in Cryptology – ASIACRYPT 2005 (Chennai, India) (Bimal K. Roy, ed.), Lecture Notes in Computer Science, vol. 3788, Springer, Berlin, Germany, December 4–8, 2005, pp. 1–20. (Cited on page 19.)

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters, *A framework for efficient and composable oblivious transfer*, Advances in Cryptology – CRYPTO 2008 (Santa Barbara, CA, USA) (David Wagner, ed.), Lecture Notes in Computer Science, vol. 5157, Springer, Berlin, Germany, August 17–21, 2008, pp. 554–571. (Cited on page 16, 17, 110, 113, 118, 128, 137.)

[PW08] Chris Peikert and Brent Waters, *Lossy trapdoor functions and their applications*, 40th Annual ACM Symposium on Theory of Computing (Victoria, British Columbia, Canada) (Richard E. Ladner and Cynthia Dwork, eds.), ACM Press, May 17–20, 2008, pp. 187–196. (Cited on page 16, 110, 114, 123, 124, 125, 126, 128.)

[RS91] Charles Rackoff and Daniel R. Simon, *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack*, Advances in Cryptology – CRYPTO'91 (Santa Barbara, CA, USA) (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, Berlin, Germany, August 11–15, 1991, pp. 433–444. (Cited on page 7, 72, 118.)

[RS09] Alon Rosen and Gil Segev, *Chosen-ciphertext security via correlated products*, TCC 2009: 6th Theory of Cryptography Conference (Omer Reingold, ed.), Lecture Notes in Computer Science, vol. 5444, Springer, Berlin, Germany, March 15–17, 2009, pp. 419–436. (Cited on page 126, 127, 139.)

[RS10] Markus Rückert and Dominique Schröder, *Fair partially blind signatures*, AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa (Stellenbosch, South Africa) (Daniel J. Bernstein and Tanja Lange, eds.), Lecture Notes in Computer Science, vol. 6055, Springer, Berlin, Germany, May 3–6, 2010, pp. 34–51. (Cited on page 39, 178.)

[RY07] Thomas Ristenpart and Scott Yilek, *The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks*, Advances in Cryptology – EUROCRYPT 2007 (Barcelona, Spain) (Moni Naor, ed.), Lecture Notes in Computer Science, vol. 4515, Springer, Berlin, Germany, May 20–24, 2007, pp. 228–245. (Cited on page 92.)

[Sah99] Amit Sahai, *Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security*, 40th Annual Symposium on Foundations of Computer Science (New York, New York, USA), IEEE Computer Society Press, October 17–19, 1999, pp. 543–553. (Cited on page 15, 51, 118, 141, 142, 295.)

[Sak00] Kazue Sako, *An auction protocol which hides bids of losers*, PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography (Melbourne, Victoria, Australia) (Hideki Imai and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1751, Springer, Berlin, Germany, January 18–20, 2000, pp. 422–432. (Cited on page 18.)

[SB04] Michael Scott and Paulo S. L. M. Barreto, *Compressed pairings*, Advances in Cryptology – CRYPTO 2004 (Santa Barbara, CA, USA) (Matthew Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer, Berlin, Germany, August 15–19, 2004, pp. 140–156. (Cited on page 204.)

[SC12] Jae Hong Seo and Jung Hee Cheon, *Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures*, TCC 2012: 9th Theory of Cryptography Conference (Taormina, Sicily, Italy) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 7194, Springer, Berlin, Germany, March 19–21, 2012, pp. 133–150. (Cited on page 37, 39, 150.)

[Sch91] Claus-Peter Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology **4** (1991), no. 3, 161–174. (Cited on page 48, 95.)

[SCWL07] Jun Shao, Zhenfu Cao, Licheng Wang, and Xiaohui Liang, *Proxy re-signature schemes without random oracles*, Progress in Cryptology - INDOCRYPT 2007: 8th International Conference in Cryptology in India (Chennai, India) (K. Srinathan, C. Pandu Rangan, and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4859, Springer, Berlin, Germany, December 9–13, 2007, pp. 197–209. (Cited on page 26, 89.)

[Seo12] Jae Hong Seo, *On the (im)possibility of projecting property in prime-order setting*, Advances in Cryptology – ASIACRYPT 2012 (Beijing, China) (Xiaoyun Wang and Kazue Sako, eds.), Lecture Notes in Computer Science, vol. 7658, Springer, Berlin, Germany, December 2–6, 2012, pp. 61–79. (Cited on page 55.)

[Sha84] Adi Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology – CRYPTO'84 (Santa Barbara, CA, USA) (G. R. Blakley and David Chaum, eds.), Lecture Notes in Computer Science, vol. 196, Springer, Berlin, Germany, August 19–23, 1984, pp. 47–53. (Cited on page 22, 24, 70.)

[Sha07] Hovav Shacham, *A Cramer-Shoup encryption scheme from the Linear Assumption and from progressively weaker Linear variants*, Cryptology ePrint Archive, Report 2007/074, 2007, `http://eprint.iacr.org/`. (Cited on page 223, 224, 251, 265.)

[Sho01] Victor Shoup, *A proposal for the ISO standard for public-key encryption (version 2.1)*, Manuscript, 2001, `http://shoup.net/`. (Cited on page 76.)

[Sho02] ———, *OAEP reconsidered*, Journal of Cryptology **15** (2002), no. 4, 223–249. (Cited on page 222.)

[Sim83] Gustavus J. Simmons, *The prisoners' problem and the subliminal channel*, Advances in Cryptology – CRYPTO'83 (Santa Barbara, CA, USA) (David Chaum, ed.), Plenum Press, New York, USA, 1983, pp. 51–67. (Cited on page 18.)

[Son01] Dawn Xiaodong Song, *Practical forward secure group signature schemes*, ACM CCS 01: 8th Conference on Computer and Communications Security (Philadelphia, PA, USA), ACM Press, November 5–8, 2001, pp. 225–234. (Cited on page 34, 196.)

[SPC95] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch, *Fair blind signatures*, Advances in Cryptology – EUROCRYPT'95 (Saint-Malo, France) (Louis C. Guillou and Jean-Jacques Quisquater, eds.), Lecture Notes in Computer Science, vol. 921, Springer, Berlin, Germany, May 21–25, 1995, pp. 209–219. (Cited on page 39, 177, 178.)

[SU12] Dominique Schröder and Dominique Unruh, *Security of blind signatures revisited*, PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography (Darmstadt, Germany) (Marc Fischlin, Johannes Buchmann, and Mark Manulis, eds.), Lecture Notes in Computer Science, vol. 7293, Springer, Berlin, Germany, May 21–23, 2012, pp. 662–679. (Cited on page 10, 149.)

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan, *Fully homomorphic encryption over the integers*, Advances in Cryptology – EUROCRYPT 2010 (French Riviera) (Henri Gilbert, ed.), Lecture Notes in Computer Science, vol. 6110, Springer, Berlin, Germany, May 30 – June 3, 2010, pp. 24–43. (Cited on page 55.)

[Ver11] Damien Vergnaud, *Efficient and secure generalized pattern matching via fast fourier transform*, AFRICACRYPT 11: 4th International Conference on Cryptology in Africa (Dakar, Senegal) (Abderrahmane Nitaj and David Pointcheval, eds.), Lecture Notes in Computer Science, vol. 6737, Springer, Berlin, Germany, July 5–7, 2011, pp. 41–58. (Cited on page 3.)

[vSN92] Sebastiaan H. von Solms and David Naccache, *On blind signatures and perfect crimes*, Computers & Security **11** (1992), no. 6, 581–583. (Cited on page 34.)

[Wat05] Brent R. Waters, *Efficient identity-based encryption without random oracles*, Advances in Cryptology – EUROCRYPT 2005 (Aarhus, Denmark) (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, Berlin, Germany, May 22–26, 2005, pp. 114–127. (Cited on page 9, 20, 26, 37, 52, 82, 89, 99, 150, 151, 159, 162, 163, 169, 202, 210, 223, 231, 254, 265, 296.)

[YY05] Adam L. Young and Moti Yung, *Questionable encryption and its applications*, Progress in Cryptology - Mycrypt 2005 (Ed Dawson and Serge Vaudenay, eds.), Lecture Notes in Computer Science, vol. 3715, Springer, 2005, pp. 210–221. (Cited on page 110, 139.)

[Zha07] Rui Zhang, *Tweaking TBE/IBE to PKE transforms with chameleon hash functions*, ACNS 07: 5th International Conference on Applied Cryptography and Network Security (Zhuhai, China) (Jonathan Katz and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4521, Springer, Berlin, Germany, June 5–8, 2007, pp. 323–339. (Cited on page 121, 122.)

[ZL06a] Sujing Zhou and Dongdai Lin, *Shorter verifier-local revocation group signatures from bilinear maps*, Cryptology ePrint Archive, Report 2006/286, 2006, `http://eprint.iacr.org/2006/286`. (Cited on page 197.)

[ZL06b] ———, *Shorter verifier-local revocation group signatures from bilinear maps*, CANS 06: 5th International Conference on Cryptology and Network Security (Suzhou, China) (David Pointcheval, Yi Mu, and Kefei Chen, eds.), Lecture Notes in Computer Science, vol. 4301, Springer, Berlin, Germany, December 8–10, 2006, pp. 126–143. (Cited on page 197.)

Abstract

This habilitation thesis presents the research work, related to the design and analysis of primitives and protocols in public-key cryptography, done by the author since his doctorate thesis. All cryptographic protocols presented in this document are analyzed in the framework of "reductionist security". We tried to minimize the use of the random oracle model and most protocols are proven secure under classical assumptions in the standard model.

Depending on the application, malleability in cryptography can be viewed as either a flaw or — especially if understood and restricted — a feature. We first present several applications of malleability for encryption schemes and signature schemes. We propose constructions achieving strong security guarantees (e.g. chosen-ciphertext security in the selective-opening setting) or new functionnalities (e.g. in the context of proxy re-cryptography). We also introduce new primitives that found applications when implemented with a suitable malleable proof system.

In a zero-knowledge proof system, a prover convinces a verifier via an interactive protocol that a mathematical statement is true, without revealing anything else than the validity of the assertion. In 2008, Groth and Sahai proposed a way to produce efficient and practical non-interactive zero-knowledge proofs for algebraic statements related to groups equipped with a bilinear map. We use the fact Groth-Sahai proofs are inherently malleable to present several privacy-preserving authentication protocols (e.g. group signatures, blind signatures, anonymous credentials, e-cash, e-voting). We introduce new specific design techniques and we use our new primitives to present efficient protocols via a modular design.

Smooth projective hashing was introduced by Cramer and Shoup in 2002. It can be seen as a weakened notion of a non-interactive zero-knowledge proof, where only a single designated verifier, who possesses some trapdoor-key, can verify proofs. We present new techniques based on smooth projective hashing for the design of commitment schemes, special signature schemes and authenticated key exchanges. In order to do so, we significantly widen the set of languages manageable via smooth projective hashing.

Résumé

Cette thèse d'habilitation présente le travail de recherche, lié à la conception et à l'analyse de primitives et protocoles en cryptographie à clé publique, effectué par l'auteur depuis sa thèse de doctorat. Tous les protocoles cryptographiques présentés dans ce document sont analysés dans le cadre de la sécurité réductionniste. Nous avons essayé de minimiser l'usage du modèle de l'oracle aléatoire et la plupart des protocoles sont prouvés sûrs sous des hypothèses classiques dans le modèle standard.

Suivant les applications, la malléabilité en cryptographie peut être vue comme une faille ou une fonctionnalité – en particulier lorsqu'elle est bien comprise et maîtrisée. Nous présentons tout d'abord plusieurs applications de la malléabilité pour des protocoles de chiffrement et de signature. Nous proposons des constructions qui atteignent des niveaux de sécurité très forts (par exemple, la sécurité adaptative pour le chiffrement dans le contexte de l'ouverture sélective de chiffrés) ou des nouvelles fonctionnalités (notamment en cryptographie délégable). Nous introduisons également de nouvelles primitives qui ont des applications lorsqu'elles sont mises en œuvre avec un système de preuve malléable.

Dans un système de preuve à divulgation nulle de connaissance, un prouveur convainc un vérifieur (via un protocole interactif) de la validité d'un énoncé mathématique, sans révéler d'autre information que cette validité. En 2008, Groth et Sahai ont proposé un système pour construire des preuves non-interactives à divulgation nulle de connaissance pour des énoncés algébriques dans des groupes dotés d'une application bilinéaire. Nous utilisons le fait que le système de preuve de Groth-Sahai est malléable pour présenter plusieurs protocoles d'authentification protégeant la vie privée des utilisateurs (par exemple, des signatures de groupe, des signatures en blanc, des accréditations anonymes, pour la monnaie électronique ou le vote électronique). Nous introduisons des techniques de conceptions spécifiques et nous utilisons nos primitives pour présenter des protocoles efficaces par une approche modulaire.

Le hachage projective lisse (*smooth projective hashing*) a été introduit par Cramer et Shoup en 2002. Il peut être vu comme une version affaiblie d'une preuve non-interactive à divulgation nulle de connaissance où un unique vérifieur, qui possède une information secrète, est capable de vérifier la validité d'une preuve. Nous présentons des nouvelles techniques basées sur le hachage projective lisse pour la conception de protocoles de mise en gage, de signatures spéciales et d'échanges de clés authentifiés. Pour cela, nous étendons significativement l'ensemble des langages traitables avec un hachage projective lisse.