This is the Full Version of the Extended Abstract that appears in
Advances in Cryptology — Proceedings of ASIACRYPT'2013 (1 – 5 December 2013, Bangalore, India)
Springer-Verlag, LNCS 8269, pages 214–234.

2014-02-17: This version provides a more detailed description of our PAKE scheme and its proof. In particular, we describe an additional property of our SPHF-friendly commitment scheme, called *strong pseudo-randomness*, which is required for the security proof of our PAKE scheme.

SPHF-Friendly Non-Interactive Commitments

Michel Abdalla[1], Fabrice Benhamouda[1], Olivier Blazy[2], Céline Chevalier[3], and David Pointcheval[1]

[1] École Normale Supérieure, CNRS-INRIA, Paris, France
[2] Ruhr-Universität Bochum, Germany
[3] Université Panthéon-Assas, Paris, France

Abstract. In 2009, Abdalla *et al.* proposed a reasonably practical password-authenticated key exchange (PAKE) secure against adaptive adversaries in the universal composability (UC) framework. It exploited the Canetti-Fischlin methodology for commitments and the Cramer-Shoup smooth projective hash functions (SPHFs), following the Gennaro-Lindell approach for PAKE. In this paper, we revisit the notion of non-interactive commitments, with a new formalism that implies UC security. In addition, we provide a quite efficient instantiation. We then extend our formalism to SPHF-friendly commitments. We thereafter show that it allows a blackbox application to one-round PAKE and oblivious transfer (OT), still secure in the UC framework against adaptive adversaries, assuming reliable erasures and a single global common reference string, even for multiple sessions. Our instantiations are more efficient than the Abdalla *et al.* PAKE in Crypto 2009 and the recent OT protocol proposed by Choi *et al.* in PKC 2013. Furthermore, the new PAKE instantiation is the first one-round scheme achieving UC security against adaptive adversaries.

1 Introduction

Commitment schemes are one of the most fundamental primitives in cryptography, serving as a building block for many cryptographic applications such as zero-knowledge proofs [GMW91] and secure multi-party computation [GMW87]. In a typical commitment scheme, there are two main phases. In a *commit* phase, the committer computes a commitment C for some message x and sends it to the receiver. Then, in an *opening* phase, the committer releases some information δ to the receiver which allows the latter to verify that C was indeed a commitment of x. To be useful in practice, a commitment scheme should satisfy two basic security properties. The first one is *hiding*, which informally guarantees that no information about x is leaked through the commitment C. The second one is *binding*, which guarantees that the committer cannot generate a commitment C that can be successfully opened to two different messages.

Smooth Projective Hash Functions (SPHFs) were introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext-secure public-key encryption schemes. In addition to providing a more intuitive abstraction for their original public-key encryption scheme in [CS98], the notion of SPHF also enabled new efficient instantiations of their scheme under different complexity assumptions, such as quadratic residuosity. Due to its usefulness, the notion of SPHF was later extended to several other contexts, such as password-authenticated key exchange (PAKE) [GL03], oblivious transfer (OT) [Kal05,CKWZ13], and blind signatures [BPV12,BBC$^+$13].

Password-Authenticated Key Exchange (PAKE) protocols were proposed in 1992 by Bellovin and Merritt [BM92] where authentication is done using a simple password, possibly drawn from a small space subject to exhaustive search. Since then, many schemes have been proposed and studied. SPHFs have been extensively used, starting with the work of Gennaro and Lindell [GL03] which generalized an earlier construction by Katz, Ostrovsky, and Yung (KOY) [KOY01], and followed by several other works [CHK$^+$05,ACP09]. More recently, a variant of SPHFs proposed by Katz and Vaikuntanathan even allowed the construction of one-round PAKE schemes [KV11,BBC$^+$13].

The first ideal functionality for PAKE protocols in the UC framework [Can01,CK02] was proposed by Canetti *et al.* [CHK$^+$05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could

lead to a secure protocol. Though quite efficient, their protocol was not known to be secure against adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states. The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL+05] using general techniques from multi-party computation (MPC). Though conceptually simple, their solution results in quite inefficient schemes.

The first reasonably practical adaptively secure PAKE was proposed by Abdalla *et al.* [ACP09], following the Gennaro-Lindell methodology with the Canetti-Fischlin commitment [CF01]. They had to build a complex SPHF to handle the verification of such a commitment. Thus, the communication complexity was high and the protocol required four rounds. No better adaptively secure scheme has been proposed so far.

Oblivious Transfer (OT) was introduced in 1981 by Rabin [Rab81] as a way to allow a receiver to get exactly one out of k messages sent by another party, the sender. In these schemes, the receiver should be oblivious to the other values, and the sender should be oblivious to which value was received. Since then, several instantiations and optimizations of such protocols have appeared in the literature, including proposals in the UC framework [NP01,CLOS02].

More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], or low communication costs [PVW08]. The 1-out-of-2 OT scheme by Choi *et al.* [CKWZ13] based on the DDH assumption seems to be the most efficient one among those that are secure against adaptive corruptions in the CRS model with erasures. But it does not scale to 1-out-of-k OT, for $k > 2$.

1.1 Properties of Commitment Schemes

Basic Properties. In addition to the binding and hiding properties, certain applications may require additional properties from a commitment scheme. One such property is *equivocability* [Bea96], which guarantees that a commitment C can be opened in more than a single way when in possession of a certain trapdoor information. Another one is *extractability*, which allows the computation of the message x committed in C when in possession of a certain trapdoor information. Yet another property that may also be useful for cryptographic applications is *non-malleability* [DDN00], which ensures that the receiver of a unopened commitment C for a message x cannot generate a commitment for a message that is related to x.

Though commitment schemes satisfying stronger properties such as *non-malleability*, *equivocability*, and *extractability* may be useful for solving specific problems, they usually stop short of guaranteeing security when composed with arbitrary protocols. To address this problem, Canetti and Fischlin [CF01] proposed an ideal functionality for commitment schemes in the universal composability (UC) framework [Can01] which guarantees all these properties simultaneously and remain secure even under concurrent compositions with arbitrary protocols. Unfortunately, they also showed that such commitment schemes can only be realized if one makes additional setup assumptions, such as the existence of a common reference string (CRS) [CF01], random oracles [HMQ04], or secure hardware tokens [Kat07].

Equivocable and Extractable Commitments. As the work of Canetti and Fischlin [CF01], this work also aims to build *non-interactive* commitment schemes which can simultaneously guarantee *non-malleability*, *equivocability*, and *extractability* properties. To this end, we first define a new notion of commitment scheme, called $\mathtt{E}^2$-commitments, for which there exists an alternative setup algorithm, whose output is computationally indistinguishable from that of a normal setup algorithm and which outputs a common trapdoor that allows for both equivocability and extractability: this trapdoor not only allows for the extraction of a committed message, but it can also be used to create simulated commitments which can be opened to any message.

To define the security of $\mathtt{E}^2$-schemes, we first extend the security notions of standard equivocable commitments and extractable commitments to the $\mathtt{E}^2$-commitment setting: Since the use of a common trapdoor for equivocability and extractability could potentially be exploited by an adversary to break the extractability or equivocability properties of an $\mathtt{E}^2$-commitment scheme, we define stronger versions of these notions, which account for the fact that the same trapdoor is used for both extractability or equivocability. In particular, in these stronger notions, the adversary is given oracle access to the simulated commitment and extractor algorithms.

Finally, after defining the security of $\mathtt{E}^2$-schemes, we further show that these schemes remain secure even under arbitrary composition with other cryptographic protocols. More precisely, we show that any $\mathtt{E}^2$–commitment scheme which meets the strong versions of the equivocability or extraction notions is a non-interactive UC-secure (multiple) commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS.

SPHF-Friendly Commitments. In this work, we are interested in building non-interactive $\mathtt{E}^2$-commitments, to which smooth projective hash functions can be efficiently associated. Unfortunately, achieving this goal is not so easy due to the equivocability property of $\mathtt{E}^2$-commitments. To understand why, let X be the domain of an SPHF function and let L be some underlying NP language such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. A key property of these SPHF functions that makes them so useful for applications such as PAKE and OT is that, for words C in L, their values can be computed using either a *secret* hashing key hk or a *public* projected key hp together a witness w to the fact that C is indeed in L. A typical example of a language in which we are interested is the language L_x corresponding to the set of elements $\{C\}$ such that C is a valid commitment of x. Unfortunately, when commitments are equivocable, the language L_x containing the set of valid commitments of x may not be well defined since a commitment C could potentially be opened to any x. To get around this problem and be able to use SPHFs with $\mathtt{E}^2$-commitments, we show that it suffices for an $\mathtt{E}^2$-commitment scheme to satisfy two properties. The first one is the stronger version of the equivocability notion, which guarantees that equivocable commitments are computationally indistinguishable from normal commitments, even when given oracle access to the simulated commitment and extractor algorithms. The second one, which is called *robustness*, is new and guarantees that commitments generated by polynomially-bounded adversaries are perfectly binding. Finally, we say that a commitment scheme is *SPHF-friendly* if it satisfies both properties and if it admits an SPHF on the languages L_x.

1.2 Contributions

A new SPHF-friendly $\mathtt{E}^2$-commitment construction. First, we define the notion of SPHF-friendly $\mathtt{E}^2$-commitment together with an instantiation. The new construction, which is called $\mathcal{E}^2\mathcal{C}$ and described in Section 4, is inspired by the commitment schemes in [CF01,CLOS02,ACP09]. Like the construction in [ACP09], it combines a variant of the Cramer-Shoup encryption scheme (as an extractable commitment scheme) and an equivocable commitment scheme to be able to simultaneously achieve both equivocability and extractability. However, unlike the construction in [ACP09], we rely on Haralambiev's perfectly hiding commitment [Har11, Section 4.1.4], instead of the Pedersen commitment [Ped92].

Since the opening value of Haralambiev's scheme is a group element that can be encrypted in one ElGamal-like ciphertext to allow extractability, this globally leads to a better communication and computational complexity for the commitment. The former is linear in $m \cdot \mathfrak{K}$, where m is the bit-length of the committed value and $\mathfrak{K}$, the security parameter. This is significantly better than the extractable commitment construction in [ACP09] which was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m. However, we point out the latter proposals in [FLM11] are not SPHF-friendly since they are not robust.

We then show in Theorem 4 that a labeled $\mathtt{E}^2$-commitment satisfying stronger notions of equivocability and extractability is a non-interactive UC-secure commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS, and we apply this result to our new construction.

One-round adaptively secure PAKE. Second, we provide a generic construction of a one-round UC-secure PAKE from any SPHF-friendly commitment, verifying an additional property called strong pseudo-randomness. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS, as shown in Section 6. In addition to being the first one-round adaptively secure PAKE, our new scheme also enjoys a much better communication complexity than previous adaptively secure PAKE schemes. For instance, in comparison to the PAKE in [ACP09], which is currently the most efficient adaptively secure PAKE, the new

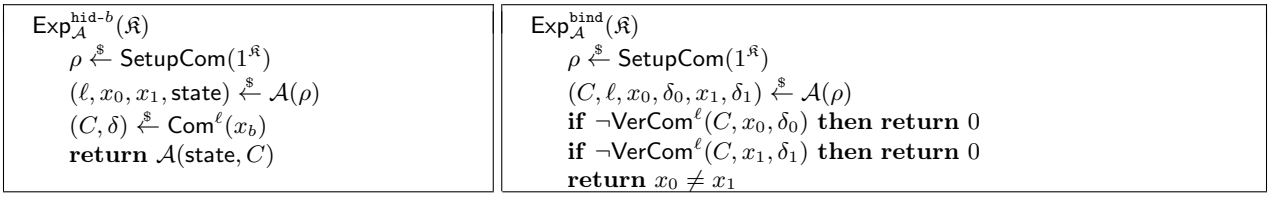

Fig. 1. Hiding and Binding Properties

scheme gains a factor of $\mathfrak{K}$ in the overall communication complexity, where $\mathfrak{K}$ is the security parameter. However, unlike their scheme, our new construction requires pairing-friendly groups.

Three-round adaptively secure 1-out-of-k OT. Third, we provide a generic construction of a three-round UC-secure 1-out-of-k OT from any SPHF-friendly commitment. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS, as shown in Section 7. Besides decreasing the total number of rounds with respect to existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solution so far [CKWZ13]. Moreover, our construction is more general and provides a solution for 1-out-of-k OT schemes while the solution in [CKWZ13] only works for $k = 2$.

Due to space restrictions, complete proofs and some details were postponed to the Appendix.

2 Basic Notions for Commitments

We first review the basic definitions of non-interactive commitments, with some examples. Then, we consider the classical additional notions of equivocability and extractability. In this paper, the qualities of adversaries will be measured by their successes and advantages in certain experiments $\mathsf{Exp}^{\mathtt{sec}}$ or $\mathsf{Exp}^{\mathtt{sec}\text{-}b}$ (between the cases $b = 0$ and $b = 1$), denoted $\mathsf{Succ}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K})$ respectively, while the security of a primitive will be measured by the maximal successes or advantages of any adversary running within a time bounded by some t in the appropriate experiments, denoted $\mathsf{Succ}^{\mathtt{sec}}(t)$ and $\mathsf{Adv}^{\mathtt{sec}}(t)$ respectively. Adversaries can keep state during the different phases. We denote $\xleftarrow{\$}$ the outcome of a probabilistic algorithm or the sampling from a uniform distribution. See Appendix A.1 for more details.

2.1 Non-Interactive Labeled Commitments

A non-interactive labeled commitment scheme $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the CRS ρ to all other algorithms;
- $\mathsf{Com}^{\ell}(x)$ takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information). This is a probabilistic algorithm;
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \perp$.

Using the experiments $\mathsf{Exp}^{\mathtt{hid}}_{\mathcal{A}}(\mathfrak{K})$ and $\mathsf{Exp}^{\mathtt{bind}}_{\mathcal{A}}(\mathfrak{K})$ defined in Figure 1, one can state the basic properties:

- *Correctness*: for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$, then $\mathsf{VerCom}^{\ell}(C, x, \delta) = 1$;
- *Hiding Property*: the commitment does not leak any information about the committed value. $\mathcal{C}$ is said (t, ε)-hiding if $\mathsf{Adv}^{\mathtt{hid}}_{\mathcal{C}}(t) \leq \varepsilon$.
- *Binding Property*: no adversary can open a commitment in two different ways. $\mathcal{C}$ is said (t, ε)-binding if $\mathsf{Succ}^{\mathtt{bind}}_{\mathcal{C}}(t) \leq \varepsilon$.

Correctness is always perfectly required, and one can also require either the binding or the hiding property to be perfect.

The reader can remark that labels are actually useless in the hiding and the binding properties. But they will become useful in $\mathtt{E}^2$-commitment schemes introduced in the next section. This is somehow similar to encryption scheme: labels are useless with encryption schemes which are just `IND-CPA`, but are very useful with `IND-CCA` encryption schemes.

2.2 Perfectly Binding Commitments: Public-Key Encryption

To get perfectly binding commitments, classical instantiations are public-key encryption schemes, which additionally provide extractability (see below). The encryption algorithm is indeed the commitment algorithm, and the random coins become the opening data that allow to check the correct procedure of the commit phase. The hiding property relies on the indistinguishability (`IND-CPA`), which is computationally achieved, whereas the binding property relies on the correctness of the encryption scheme and is perfect.

Let us define the ElGamal-based commitment scheme:

- $\mathsf{SetupCom}(1^\mathfrak{K})$ chooses a cyclic group $\mathbb{G}$ of prime order p, g a generator for this group and a random scalar $z \xleftarrow{\$} \mathbb{Z}_p$. It sets the CRS $\rho = (\mathbb{G}, g, h = g^z)$;
- $\mathsf{Com}(M)$, for $M \in \mathbb{G}$, chooses a random element $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the pair $(C = (u = g^r, e = h^r \cdot M), \delta = r)$;
- $\mathsf{VerCom}(C = (u, e), M, \delta = r)$ checks whether $C = (u = g^r, e = h^r \cdot M)$.

This commitment scheme is hiding under the DDH assumption and perfectly binding. It is even extractable using the decryption key z: $M = e/u^z$. However, it is not labeled. The Cramer-Shoup encryption scheme [CS98] admits labels and is extractable and non-malleable, thanks to the `IND-CCA` security level. It is reviewed in Appendix B, with some extensions, since we will use it later in our applications.

2.3 Perfectly Hiding Commitments

The Pedersen scheme [Ped92] is the most famous perfectly hiding commitment: $\mathsf{Com}(m) = g^m h^r$ for a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$ and a fixed basis $h \in \mathbb{G}$. The binding property relies on the DL assumption. Unfortunately, the opening value is the scalar r, which makes it hard to encrypt/decrypt efficiently, as required in our construction below. Haralambiev [Har11, Section 4.1.4] recently proposed a new commitment scheme, called TC4 (without label), with a group element as opening value:

- $\mathsf{SetupCom}(1^\mathfrak{K})$ chooses an asymmetric pairing-friendly setting $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$, with an additional independent generator $T \in \mathbb{G}_2$. It sets the CRS $\rho = (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, T, \mathbb{G}_T, p, e)$;
- $\mathsf{Com}(x)$, for $x \in \mathbb{Z}_p$, chooses a random element $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the pair $(C = g_2^r T^x, \delta = g_1^r)$;
- $\mathsf{VerCom}(C, x, \delta)$ checks whether $e(g_1, C/T^x) = e(\delta, g_2)$.

This commitment scheme is clearly perfectly hiding, since the groups are cyclic, and for any $C \in \mathbb{G}_2$, $x \in \mathbb{Z}_p$, there exists $\delta \in \mathbb{G}_1$ that satisfies $e(g_1, C/T^x) = e(\delta, g_2)$. More precisely, if $C = g_2^u$ and $T = g_2^t$, then $\delta = g_1^{u-tx}$ opens C to any x. The binding property holds under the DDH assumption in $\mathbb{G}_2$, as proven in [Har11, Section 4.1.4].

2.4 Equivocable Commitments

An equivocable commitment scheme $\mathcal{C}$ extends on the previous definition, with SetupCom, Com, VerCom, and a second setup $\mathsf{SetupComT}(1^\mathfrak{K})$ that additionally outputs a trapdoor τ, and

- $\mathsf{SimCom}^\ell(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;

$\mathsf{Exp}^{\mathtt{sim\text{-}ind}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(\ell, x, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot)}(\rho)$
if $b = 0$ **then** $(C,\delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$
else $(C,\delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$
return $\mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \delta)$

$\mathsf{Exp}^{\mathtt{bind\text{-}ext}}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(C, \ell, x, \delta) \xleftarrow{\$} \mathcal{A}^{\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
$x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
if $x' = x$ **then return** 0
else return $\mathsf{VerCom}^{\ell}(C, x, \delta)$

Fig. 2. Simulation Indistinguishability and Binding Extractability

– $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, and an equivocation key eqk for this commitment, and outputs an opening data δ for C and ℓ on x.

Let us denote SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C,\delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$. Three additional properties are then associated: a *correctness* property, and two *indistinguishability* properties, which all together imply the *hiding* property.

– *Trapdoor Correctness*: all simulated commitments can be opened on any message: for all ℓ, x, if $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$, then $\mathsf{VerCom}^{\ell}(C, x, \delta) = 1$;
– *Setup Indistinguishability*: one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$. $\mathcal{C}$ is said (t,ε)-setup-indistinguishable if the two distributions for ρ are (t,ε)-computationally indistinguishable. We denote $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$ the distance between the two distributions.
– *Simulation Indistinguishability*: one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments. $\mathcal{C}$ is said (t,ε)-simulation-indistinguishable if $\mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t) \leq \varepsilon$ (see the experiments $\mathsf{Exp}^{\mathtt{sim\text{-}ind}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$ in Figure 2).

More precisely, when the trapdoor correctness is satisfied, since commitments generated by SimCom are perfectly hiding (they can be opened in any way using $\mathsf{OpenCom}$), $\mathsf{Adv}^{\mathtt{hid}}_{\mathcal{C}}(t) \leq \mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t)$.

Definition 1 (Equivocable Commitment). *A commitment scheme $\mathcal{C}$ is said (t,ε)-equivocable if, first, the basic commitment scheme satisfies the correctness property and is both (t,ε)-binding and (t,ε)-hiding, and, secondly, the additional algorithms guarantee the trapdoor correctness and make it both (t,ε)-setup-indistinguishable and (t,ε)-simulation-indistinguishable.*
One denotes $\mathsf{Adv}^{\mathtt{equiv}}_{\mathcal{C}}(t)$ the maximum of $\mathsf{Succ}^{\mathtt{bind}}_{\mathcal{C}}(t)$, $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$, and $\mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t)$; it should be upper-bounded by ε.

2.5 Extractable Commitments

An extractable commitment scheme $\mathcal{C}$ also extends on the initial definition, with $\mathsf{SetupCom}$, Com, VerCom, as well as the second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and

– $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

As above, three additional properties are then associated: a *correctness* property, and the *setup indistinguishability*, but also an *extractability* property, which implies, together with the setup indistinguishability, the *binding* property:

– *Trapdoor Correctness*: all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C,\delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C,\tau) = x$;
– *Setup Indistinguishability*: as above;
– *Binding Extractability*: one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x. $\mathcal{C}$ is said (t,ε)-binding-extractable if $\mathsf{Succ}^{\mathtt{bind\text{-}ext}}_{\mathcal{C}}(t) \leq \varepsilon$ (see the experiment $\mathsf{Exp}^{\mathtt{bind\text{-}ext}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 2).

More precisely, when one breaks the binding property with $(C, \ell, x_0, \delta_0, x_1, \delta_1)$, if the extraction oracle outputs $x' = x_0$, then one can output (C, ℓ, x_1, δ_1), otherwise one can output (C, ℓ, x_0, δ_0). In both cases, this breaks the binding-extractability: $\mathsf{Adv}^{\mathtt{bind}}_{\mathcal{C}}(t) \leq \mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t) + \mathsf{Succ}^{\mathtt{bind\text{-}ext}}_{\mathcal{C}}(t)$.

Definition 2 (Extractable Commitment). *A commitment scheme $\mathcal{C}$ is said (t, ε)-extractable if, first, the basic commitment scheme satisfies the correctness property and is both (t, ε)-binding and (t, ε)-hiding, and, secondly, the additional algorithms guarantee the trapdoor correctness and make it both (t, ε)-setup-indistinguishable and (t, ε)-binding-extractable.*
One denotes $\mathsf{Adv}^{\mathtt{ext}}_{\mathcal{C}}(t)$ the maximum of $\mathsf{Adv}^{\mathtt{hid}}_{\mathcal{C}}(t)$, $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$, and $\mathsf{Succ}^{\mathtt{bind\text{-}ext}}_{\mathcal{C}}(t)$; it should be upper-bounded by ε.

3 Equivocable and Extractable Commitments

3.1 $\mathtt{E}^2$-Commitments: Equivocable and Extractable

Public-key encryption schemes are perfectly binding commitments that are additionally extractable. The Pedersen and Haralambiev commitments are perfectly hiding commitments that are additionally equivocable. But none of them have the two properties at the same time. This is now our goal.

Definition 3 ($\mathtt{E}^2$-Commitment). *A commitment scheme $\mathcal{C}$ is said (t, ε)-$\mathtt{E}^2$ (equivocable and extractable) if the indistinguishable setup algorithm outputs a common trapdoor that allows both equivocability and extractability. If one denotes $\mathsf{Adv}^{\mathtt{e}^2}_{\mathcal{C}}(t)$ the maximum of $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$, $\mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t)$, and $\mathsf{Succ}^{\mathtt{bind\text{-}ext}}_{\mathcal{C}}(t)$, then it should be upper-bounded by ε.*

But with such a common trapdoor, the adversary could exploit the equivocation queries to break extractability and extraction queries to break equivocability. Stronger notions can thus be defined, using the experiments $\mathsf{Exp}^{\mathtt{s\text{-}sim\text{-}ind}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$ and $\mathsf{Exp}^{\mathtt{s\text{-}bind\text{-}ext}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 3, in which SCom is supposed to store each query/answer (ℓ, x, C) in a list Λ and ExtCom-queries on such an SCom-output (ℓ, C) are answered by x (as it would be when using Com instead of SCom).

- *Strong Simulation Indistinguishability*: one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). $\mathcal{C}$ is said (t, ε)-strongly-simulation-indistinguishable if $\mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{C}}(t) \leq \varepsilon$;
- *Strong Binding Extractability* (informally introduced in [CLOS02] as "simulation extractability"): one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data (not given by SCom) to an input x while the commitment does not extract to x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). $\mathcal{C}$ is said (t, ε)-strongly-binding-extractable if $\mathsf{Succ}^{\mathtt{s\text{-}bind\text{-}ext}}_{\mathcal{C}}(t) \leq \varepsilon$.

They both imply the respective weaker notions since they just differ by giving access to the ExtCom-oracle in the former game, and to the SCom oracle in the latter. We insist that ExtCom-queries on SCom-outputs are answered by the related SCom-inputs. Otherwise, the former game would be void. In addition, VerCom always rejects inputs with $x = \perp$, which is useful in the latter game.

3.2 UC-Secure Commitments

The security definition for commitment schemes in the UC framework was presented by Canetti and Fischlin [CF01], refined by Canetti [Can05]. The ideal functionality is presented in Figure 4, where a *public delayed output* is an output first sent to the adversary $\mathcal{S}$ that eventually decides if and when the message is actually delivered to the recipient. In case of corruption of the committer, if this is before the `Receipt`-message for the receiver, the adversary chooses the committed value, otherwise it is provided by the ideal functionality, according to the `Commit`-message. Note this is actually the multiple-commitment functionality that allows multiple executions of the commitment protocol (multiple ssid's) for the same functionality instance (one sid). This avoids the use of joint-state UC [CR03].

$\mathsf{Exp}^{\texttt{s-sim-ind-}b}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$;
$(\ell, x, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
if $b = 0$ **then** $(C,\delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$
else $(C,\delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$
return $\mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \delta)$

$\mathsf{Exp}^{\texttt{s-bind-ext}}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(C, \ell, x, \delta) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
$x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
if $(\ell, x', C) \in \Lambda$ **then return** 0
if $x' = x$ **then return** 0
else return $\mathsf{VerCom}^{\ell}(C, x, \delta)$

Fig. 3. Strong Simulation Indistinguishability and Strong Binding Extractability

The functionality $\mathcal{F}_{\text{com}}$ is parametrized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

Commit phase: Upon receiving a query $(\texttt{Commit}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j}, \boldsymbol{x})$ from party $\boldsymbol{P_i}$: record the tuple $(\text{sid}, \text{ssid}, P_i, P_j, x)$ and generate a *public delayed output* $(\texttt{Receipt}, \text{sid}, \text{ssid}, P_i, P_j)$ to P_j. Ignore further `Commit`-message with the same ssid from P_i.

Decommit phase. Upon receiving a query $(\texttt{Reveal}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j})$ from party $\boldsymbol{P_i}$: ignore the message if $(\text{sid}, \text{ssid}, P_i, P_j, x)$ is not recorded; otherwise mark the record $(\text{sid}, \text{ssid}, P_i, P_j)$ as revealed and generate a *public delayed output* $(\texttt{Revealed}, \text{sid}, \text{ssid}, P_i, P_j, x)$ to P_j. Ignore further `Reveal`-message with the same ssid from P_i.

Fig. 4. Ideal Functionality for Commitment Scheme $\boldsymbol{\mathcal{F}}_{\text{com}}$

Theorem 4. *A labeled E^2-commitment scheme $\mathcal{C}$, that is in addition strongly-simulation-indistinguishable or strongly-binding-extractable, is a non-interactive UC-secure commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*
More precisely, for any environment, its advantage in distinguishing the ideal world (with the ideal functionality from Figure 4) and the real world (with the commitment scheme $\mathcal{C}$) is bounded by both $\mathsf{Adv}^{\texttt{setup-ind}}_{\mathcal{C}}(t) + q_s \cdot \mathsf{Adv}^{\texttt{sim-ind}}_{\mathcal{C}}(t) + \mathsf{Succ}^{\texttt{s-bind-ext}}_{\mathcal{C}}(t)$ and $\mathsf{Adv}^{\texttt{setup-ind}}_{\mathcal{C}}(t) + q_s \cdot \mathsf{Adv}^{\texttt{s-sim-ind}}_{\mathcal{C}}(t) + \mathsf{Succ}^{\texttt{bind-ext}}_{\mathcal{C}}(t)$, where q_s is the number of concurrent sessions and t its running time.

Proof. The full proof with the cost of the reduction are given in Appendix D, but let us provide here the simulator:

- when receiving a commitment C from the adversary, and thus either freshly generated by the adversary or a replay of a commitment C generated by the simulator in another session (with a different label), the simulator extracts the committed value x, and uses it to send a `Commit` message to the ideal functionality. A dummy value is used in case of bad extraction;
- when receiving a `Receipt`-message, which means that an honest player has committed a value, the simulator generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\text{sid}, \text{ssid}, P_i, P_j)$, to send C during the commit phase of the honest player;
- when receiving (x, δ), if the verification succeeds, the simulator asks for a `Reveal` query to the ideal functionality;
- when receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment.

Any corruption just reveals x earlier, which allows a correct simulation of the opening. □

4 A Construction of Labeled E^2-Commitment Scheme

4.1 Labeled Cramer-Shoup Encryption on Vectors

For our construction we use a variant of the Cramer-Shoup encryption scheme for vectors of messages. Let $\mathbb{G}$ be a cyclic group of order p, with two independent generators g and h. The secret decryption key is a random vector $\mathsf{sk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$ and the public encryption key is $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$, where H is randomly chosen in a collision-resistant hash function family $\mathcal{H}$ (actually, second-preimage

resistance is enough). For a message-vector $\boldsymbol{M} = (M_i)_{i=1,\ldots,m} \in \mathbb{G}^m$, the multi-Cramer-Shoup encryption is defined as $m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{M}; (r_i)_i) = (\mathsf{CS}^{\ell}_{\mathsf{pk}}(M_i, \theta; r_i) = (u_i = g^{r_i}, v_i = h^{r_i}, e_i = f^{r_i} \cdot M_i, w_i = (cd^{\theta})^{r_i}))_i$, where $\theta = H(\ell, (u_i, v_i, e_i)_i)$ is the same for all the w_i's to ensure non-malleability contrary to what we would have if we had just concatenated Cramer-Shoup ciphertexts of the M_i's. Such a ciphertext $C = (u_i, v_i, e_i, w_i)_i$ is decrypted by $M_i = e_i/u_i^z$, after having checked the validity of the ciphertext, $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, for $i = 1, \ldots, m$. This multi-Cramer-Shoup encryption scheme, denoted MCS, is $\mathtt{IND\text{-}CCA}$ under the DDH assumption. It even verifies a stronger property $\mathtt{VIND\text{-}PO\text{-}CCA}$ (for Vector-Indistinguishability with Partial Opening under Chosen-Ciphertext Attacks), where the random coins r_i, of the common coordinates in the two challenge vectors, are published with the challenge ciphertext (partial-opening of the random coins, see Appendix B for more details). This will be useful for the security proof of our commitment $\mathcal{E}^2\mathcal{C}$ (see below):

$$\mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathsf{MCS}}(m, q_d, \gamma, t) \leq 2\gamma \cdot \mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t) + \frac{m(\gamma+1)q_d}{p}, \tag{1}$$

where m is the length of the encrypted vectors, q_d is the maximal number of decryption queries, and γ the bound on the number of distinct components in the two challenge vectors; and where $\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}$ and $\mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t)$ are respectively the maximum advantage of an adversary against the DDH problem and the maximum probability for an adversary to find a collision for $H \stackrel{\$}{\leftarrow} \mathcal{H}$, in time t.

4.2 Construction

In this section, we provide a concrete construction $\mathcal{E}^2\mathcal{C}$, inspired from [CF01,CLOS02,ACP09], with the above multi-Cramer-Shoup encryption (as an extractable commitment scheme) and the TC4 Haralambiev's equivocable commitment scheme [Har11, Section 4.1.4]. The latter will allow equivocability while the former will provide extractability:

- $\mathsf{SetupComT}(1^{\mathfrak{K}})$ generates a pairing-friendly setting $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$, with another independent generator h_1 of $\mathbb{G}_1$. It then generates the parameters of a Cramer-Shoup-based commitment in $\mathbb{G}_1$: $x_1, x_2, y_1, y_2, z \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and $H \stackrel{\$}{\leftarrow} \mathcal{H}$, and sets $\mathsf{pk} = (g_1, h_1, c = g_1^{x_1} h_1^{x_2}, d = g_1^{y_1} h_1^{y_2}, f_1 = g_1^z, H)$. It then chooses a random scalar $t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, and sets $T = g_2^t$. The CRS ρ is set as (pk, T) and the trapdoor τ is the decryption key (x_1, x_2, y_1, y_2, z) (a.k.a. extraction trapdoor) together with t (a.k.a. equivocation trapdoor). For $\mathsf{SetupCom}(1^{\mathfrak{K}})$, the CRS is generated the same way, but forgetting the scalars, and thus without any trapdoor;
- $\mathsf{Com}^{\ell}(\boldsymbol{M})$, for $\boldsymbol{M} = (M_i)_i \in \{0,1\}^m$ and a label ℓ, works as follows:
  - For $i = 1, \ldots, m$, it chooses a random scalar $r_{i,M_i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, sets $r_{i,1-M_i} = 0$, and commits to M_i, using the TC4 commitment scheme with r_{i,M_i} as randomness: $a_i = g_2^{r_{i,M_i}} T^{M_i}$, and sets $d_{i,j} = g_1^{r_{i,j}}$ for $j = 0, 1$, which makes d_{i,M_i} the opening value for a_i to M_i; Let us also write $\boldsymbol{a} = (a_1, \ldots, a_m)$, the tuple of commitments.
  - For $i = 1, \ldots, m$ and $j = 0, 1$, it gets $\boldsymbol{b} = (b_{i,j})_{i,j} = 2m\text{-}\mathsf{MCS}^{\ell'}_{\mathsf{pk}}(\boldsymbol{d}; \boldsymbol{s})$, that is $(u_{i,j}, v_{i,j}, e_{i,j}, w_{i,j})_{i,j}$, where $\boldsymbol{d} = (d_{i,j})_{i,j}$ computed above, $\boldsymbol{s} = (s_{i,j})_{i,j} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{2m}$, and $\ell' = (\ell, \boldsymbol{a})$.

  The commitment is $C = (\boldsymbol{a}, \boldsymbol{b})$, and the opening information is the m-tuple $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$.
- $\mathsf{VerCom}^{\ell}(C, \boldsymbol{M}, \delta)$ checks the validity of the ciphertexts b_{i,M_i} with s_{i,M_i} and θ computed on the full ciphertext C, extracts d_{i,M_i} from b_{i,M_i} and s_{i,M_i}, and checks whether $e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2)$, for $i = 1, \ldots, m$.
- $\mathsf{SimCom}^{\ell}(\tau)$ takes as input the equivocation trapdoor, namely t, and outputs $C = (\boldsymbol{a}, \boldsymbol{b})$ and $\mathsf{eqk} = \boldsymbol{s}$, where
  - For $i = 1, \ldots, m$, it chooses a random scalar $r_{i,0} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, sets $r_{i,1} = r_{i,0} - t$, and commits to both 0 and 1, using the TC4 commitment scheme with $r_{i,0}$ and $r_{i,1}$ as respective randomness: $a_i = g_2^{r_{i,0}} = g_2^{r_{i,1}} T$, and $d_{i,j} = g_1^{r_{i,j}}$ for $j = 0, 1$, which makes $d_{i,j}$ the opening value for a_i to the value $j \in \{0, 1\}$. This leads to $\boldsymbol{a}$;
  - $\boldsymbol{b}$ is built as above: $\boldsymbol{b} = (b_{i,j})_{i,j} = 2m\text{-}\mathsf{MCS}^{\ell'}_{\mathsf{pk}}(\boldsymbol{d}; \boldsymbol{s})$, with random scalars $(s_{i,j})_{i,j}$.

- $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$ simply extracts the useful values from $\mathsf{eqk} = \boldsymbol{s}$ to make the opening value $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$ in order to open to $\boldsymbol{M} = (M_i)_i$.
- $\mathsf{ExtCom}^{\ell}(\tau, C)$ takes as input the extraction trapdoor, namely the Cramer-Shoup decryption key. Given $\boldsymbol{b}$, it can decrypt all the $b_{i,j}$ into $d_{i,j}$ and check whether $e(g_1, a_i/T^j) = e(d_{i,j}, g_2)$ or not. If, for each i, exactly one $j = M_i$ satisfies the equality, then the extraction algorithm outputs $(M_i)_i$, otherwise (no correct decryption or ambiguity with several possibilities) it outputs $\perp$.

4.3 Security Properties

The above commitment scheme $\mathcal{E}^2\mathcal{C}$ is a labeled $\mathtt{E}^2$-commitment, with both strong-simulation-indistinguishability and strong-binding-extractability, under the DDH assumptions in both $\mathbb{G}_1$ and $\mathbb{G}_2$. It is thus a UC-secure commitment scheme. The stronger `VIND-PO-CCA` security notion for the encryption scheme is required because the SCom/Com oracle does not only output the commitment (and thus the ciphertexts) but also the opening values which include the random coins of the encryption, but just for the plaintext components that are the same in the two vectors, since the two vectors only differ for unnecessary data (namely the $d_{i,1-M_i}$'s) in the security proof. More details can be found in the full proof, in Appendix D, which leads to $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) = 0$, $\mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) \leq \mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathsf{MCS}}(2m, q_d, m, t)$, and $\mathsf{Succ}^{\mathtt{s\text{-}bind\text{-}ext}}_{\mathcal{E}^2\mathcal{C}}(t) \leq q_c \cdot \mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}_2}(t)$, where q_c is the number of SCom-queries and q_d the number of ExtCom-queries.

5 SPHF-Friendly Commitments

5.1 Smooth Projective Hash Functions

Projective hash function families were first introduced by Cramer and Shoup [CS02], but we here use the definitions of Gennaro and Lindell [GL03], provided to build secure password-based authenticated key exchange protocols, together with non-malleable commitments.

Let X be the domain of these functions and let L be a certain subset of this domain (a language). A key property of these functions is that, for words C in L, their values can be computed by using either a *secret* hashing key hk or a *public* projection key hp but with a witness w of the fact that C is indeed in L:

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

The set of hash values is called the *range* of the SPHF and is denoted Π. The *correctness* of the SPHF assures that if $C \in L$ with w a witness of this fact, then $\mathsf{Hash}(\mathsf{hk}, L, C) = \mathsf{ProjHash}(\mathsf{hp}, L, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin L$, $\mathsf{Hash}(\mathsf{hk}, L, C)$ is *statistically* indistinguishable from a random element, even knowing hp.

Note that HashKG and ProjKG can just depend partially on L (a superset L') and not at all on C: we then note $\mathsf{HashKG}(L')$ and $\mathsf{ProjKG}(\mathsf{hk}, L', \perp)$ (see [BBC+13] for more details on GL-SPHF and KV-SPHF and language definitions).

5.2 Robust Commitments

For a long time, SPHFs have been used to implicitly check some statements, on language membership, such as "C indeed encrypts x". This easily extends to perfectly binding commitments with labels: $L_x = \{(\ell, C) | \exists \delta, \mathsf{VerCom}^{\ell}(C, x, \delta) = 1\}$. But when commitments are equivocable, this intuitively means that a commitment C with the label ℓ contains any x and is thus in all the languages L_x. In order to be able to use SPHFs with $\mathtt{E}^2$-commitments, we want the commitments generated by polynomially-bounded adversaries to be perfectly binding, and thus to belong to at most one language L_x. We thus need a *robust verification* property for such $\mathtt{E}^2$*-commitments*.

$\mathsf{Exp}^{\mathtt{robust}}_{\mathcal{A}}(\mathfrak{K})$
$(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(C, \ell) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}'(\tau,\cdot),\mathsf{ExtCom}'(\tau,\cdot)}(\rho)$
$x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
if $(\ell, x', C) \in \Lambda$ **then return** 0
if $\exists x \neq x', \exists \delta, \mathsf{VerCom}^{\ell}(C, x, \delta)$ **then return** 1
else return 0

Fig. 5. Robustness

Definition 5 (Robustness). *One cannot produce a commitment and a label that extracts to x' (possibly $x' = \perp$) such that there exists a valid opening data to a different input x, even with oracle access to the extraction oracle (*ExtCom*) and to fake commitments (using* SCom*). $\mathcal{C}$ is said (t, ε)-robust if $\mathsf{Succ}^{\mathtt{robust}}_{\mathcal{C}}(t) \leq \varepsilon$, according to the experiment $\mathsf{Exp}^{\mathtt{robust}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 5.*

It is important to note that the latter experiment $\mathsf{Exp}^{\mathtt{robust}}_{\mathcal{A}}(\mathfrak{K})$ may not be run in polynomial time. Robustness implies strong-binding-extractability.

5.3 Properties of SPHF-Friendly Commitments

We are now ready to define SPHF-friendly commitments, which admit an SPHF on the languages $L_x = \{(\ell, C) | \exists \delta, \mathsf{VerCom}^{\ell}(C, x, \delta) = 1\}$, and to discuss about them:

Definition 6 (SPHF-Friendly Commitments). *An SPHF-friendly commitment is an E^2-commitment that admits an SPHF on the languages L_x, and that is both strongly-simulation-indistinguishable and robust.*

Let us consider such a family $\mathcal{F}$ of SPHFs on languages L_x for $x \in X$, with X a non trivial set (with at least two elements), with hash values in the set G. From the smoothness of the SPHF on L_x, one can derive the two following properties on SPHF-friendly commitments, modeled by the experiments in Figure 6. The first notion of *smoothness* deals with adversary-generated commitments, that are likely perfectly binding from the robustness, while the second notion of *pseudo-randomness* deals with simulated commitments, that are perfectly hiding. They are inspired by the security games from [GL03].

In both security games, note that when hk and hp do not depend on x nor on C, and when the smoothness holds even if the adversary can choose C after having seen hp (*i.e.*, the SPHF is actually a KV-SPHF [BBC+13]), they can be generated from the beginning of the games, with hp given to the adversary much earlier.

Smoothness of SPHF-Friendly Commitments. If the adversary $\mathcal{A}$, with access to the oracles SCom and ExtCom, outputs a fresh commitment (ℓ, C) that extracts to $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$, then the robustness guarantees that for any $x \neq x'$, $(\ell, C) \notin L_x$ (excepted with small probability), and thus the distribution of the hash value is statistically indistinguishable from the random distribution, even when knowing hp. In the experiment $\mathsf{Exp}^{\mathtt{c\text{-}smooth}}_{\mathcal{A}}(\mathfrak{K})$, we let the adversary choose x, and we have: $\mathsf{Adv}^{\mathtt{c\text{-}smooth}}_{\mathcal{C},\mathcal{F}}(t) \leq \mathsf{Succ}^{\mathtt{robust}}_{\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{smooth}}_{\mathcal{F}}$.

Pseudo-Randomness of SPHF on Robust Commitments. If the adversary $\mathcal{A}$ is given a commitment C by SimCom with label ℓ, adversary-chosen, even with access to the oracles SCom and ExtCom, then for any x, it cannot distinguish the hash value of (ℓ, C) on language L_x from a random value, even being given hp, since C could have been generated as $\mathsf{Com}^{\ell}(x'')$ for some $x'' \neq x$, which excludes it to belong to L_x, under the robustness. In the experiment $\mathsf{Exp}^{\mathtt{c\text{-}ps\text{-}rand}}_{\mathcal{A}}(\mathfrak{K})$, we let the adversary choose (ℓ, x), and we have: $\mathsf{Adv}^{\mathtt{c\text{-}ps\text{-}rand}}_{\mathcal{C},\mathcal{F}}(t) \leq \mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{C}}(t) + \mathsf{Succ}^{\mathtt{robust}}_{\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{smooth}}_{\mathcal{F}}$.

$\mathsf{Exp}^{\texttt{c-smooth-}b}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(C,\ell,x,\mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\rho)$; $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau,C)$
if $(\ell,x',C) \in \Lambda$ **then return** 0
$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(L_x)$; $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},L_x,(\ell,C))$
if $b=0 \vee x'=x$ **then** $H \leftarrow \mathsf{Hash}(\mathsf{hk},L_x,(\ell,C))$
else $H \xleftarrow{\$} \Pi$
return $\mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\mathsf{state},\mathsf{hp},H)$

$\mathsf{Exp}^{\texttt{c-ps-rand-}b}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(\ell,x,\mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\rho)$; $C \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$
$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(L_x)$; $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},L_x,(\ell,C))$
if $b=0$ **then** $H \leftarrow \mathsf{Hash}(\mathsf{hk},L_x,(\ell,C))$
else $H \xleftarrow{\$} \Pi$
return $\mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\mathsf{state},C,\mathsf{hp},H)$

$\mathsf{Exp}^{\texttt{c-s-ps-rand-}b}_{\mathcal{A}}(\mathfrak{K})$
$(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
$(\ell,x,\mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\rho)$; $C \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$
$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(L_x)$; $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},L_x,\perp)$
if $b=0$ **then** $H \leftarrow \mathsf{Hash}(\mathsf{hk},L_x,(\ell,C))$
else $H \xleftarrow{\$} \Pi$
$(\ell',C',\mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(\mathsf{state},C,\mathsf{hp},H)$
if $(\ell',?,C') \in \Lambda$ **then** $H' \leftarrow \perp$
else $H' \leftarrow \mathsf{Hash}(\mathsf{hk},L_x,(\ell',C')$
return $\mathcal{A}^{\mathsf{SCom}(\tau,\cdot),\mathsf{ExtCom}(\tau,\cdot)}(H')$

Fig. 6. Smoothness, Pseudo-Randomness and Strong Pseudo-Randomness

Strong Pseudo-Randomness. Besides these two properties which hold for any SPHF-friendly commitment, we need a third property for our one-round PAKE protocol. This property, called strong pseudo-randomness, is defined by the experiment $\mathsf{Exp}^{\texttt{c-s-ps-rand}}_{\mathcal{A}}(\mathfrak{K})$ depicted in Figure 6. It is a strong version of the pseudo-randomness where the adversary is also given the hash value of a commitment of its choice (obviously not generated by SCom or SimCom though, hence the test with Λ which also contains (C,ℓ,x)). This property only makes sense when the projection key does not depend on the word C to be hashed. It thus applies to KV-SPHF only.

5.4 Our Commitment Scheme $\mathcal{E}^2\mathcal{C}$ is SPHF-Friendly

In order to be *SPHF-friendly*, the commitment first needs to be *strongly-simulation-indistinguishable* and *robust*. We have already shown the former property, and the latter is also proven in Appendix D. One additionally needs an SPHF able to check the verification equation: using the notations from Section 4.2, $C=(\boldsymbol{a},\boldsymbol{b})$ is a commitment of $\boldsymbol{M}=(M_i)_i$, if there exist $\delta=(s_{1,M_1},\ldots,s_{m,M_m})$ and $(d_{1,M_1},\ldots,d_{m,M_m})$ such that $b_{i,M_i}=(u_{i,M_i},v_{i,M_i},e_{i,M_i},w_{i,M_i})=\mathsf{CS}^{\ell'}_{\mathsf{pk}}(d_{i,M_i},\theta;s_{i,M_i})$ (with a particular θ) and $e(g_1,a_i/T^{M_i})=e(d_{i,M_i},g_2)$, for $i=1,\ldots,m$. Since e is non-degenerated, we can eliminate the need of d_{i,M_i}, by lifting everything in $\mathbb{G}_T$, and checking that, first, the ciphertexts are all valid:

$$e(u_{i,M_i},g_2)=e(g_1^{s_{i,M_i}},g_2) \qquad e(v_{i,M_i},g_2)=e(h_1^{s_{i,M_i}},g_2) \qquad e(w_{i,M_i},g_2)=e((cd^{\theta})^{s_{i,M_i}},g_2)$$

and, second, the plaintexts satisfy the appropriate relations:

$$e(e_{i,M_i},g_2)=e(f_1^{s_{i,M_i}},g_2)\cdot e(g_1,a_i/T^{M_i}).$$

Table 1. Comparison with existing non-interactive UC-secure commitments with a single global CRS (m = bit-length of the committed value, $\mathfrak{K}$ = security parameter)

	SPHF-Friendly	Commitment C	Decommitment δ	Assumption
[ACP09][a]	yes	$(m + 16m\mathfrak{K}) \times \mathbb{G}$	$2m\mathfrak{K} \times \mathbb{Z}_p$	DDH
[FLM11], 1	no	$5 \times \mathbb{G}$	$16 \times \mathbb{G}$	DLIN
[FLM11], 2	no	$37 \times \mathbb{G}$	$3 \times \mathbb{G}$	DLIN
this paper	yes	$8m \times \mathbb{G}_1 + m \times \mathbb{G}_2$	$m \times \mathbb{Z}_p$	SXDH

[a] slight variant without one-time signature but using labels for the IND-CCA security of the multi-Cramer-Shoup ciphertexts, as in our new scheme, and supposing that an element in the cyclic group $\mathbb{G}$ has size $2\mathfrak{K}$, to withstand generic attacks.

From these expressions we derive several constructions of such SPHFs in Appendix C, and focus here on the most interesting ones for the following applications:

– First, when C is sent in advance (known when generating hp), as in the OT protocol described in Section 7, for $\mathsf{hk} = (\eta, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^5$, and $\mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^{\eta} h_1^{\alpha} f_1^{\beta} (cd^{\theta})^{\mu}) \in \mathbb{Z}_p \times \mathbb{G}_1$:

$$\begin{aligned} H &= \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \\ &\stackrel{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^{\eta} \cdot v_{i,M_i}^{\alpha}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta} \cdot e(w_{i,M_i}^{\mu}, g_2) \right)^{\varepsilon^{i-1}} \\ &= e(\prod_i \mathsf{hp}_1^{s_{i,M_i} \varepsilon^{i-1}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, \delta) = H'. \end{aligned}$$

– Then, when C is not necessarily known for computing hp, as in the one-round PAKE, described in Section 6, for $\mathsf{hk} = (\eta_{i,1}, \eta_{i,2}, \alpha_i, \beta_i, \mu_i)_i \xleftarrow{\$} \mathbb{Z}_p^{5m}$, and $\mathsf{hp} = (\mathsf{hp}_{i,1} = g_1^{\eta_{i,1}} h_1^{\alpha_i} f_1^{\beta_i} c^{\mu_i}, \mathsf{hp}_{i,2} = g_1^{\eta_{i,2}} d^{\mu_i})_i \in \mathbb{G}_1^{2m}$:

$$\begin{aligned} H &= \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \\ &\stackrel{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^{(\eta_{i,1} + \theta \eta_{i,2})} \cdot v_{i,M_i}^{\alpha_i}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta_i} \cdot e(w_{i,M_i}^{\mu_i}, g_2) \right) \\ &= e(\prod_i (\mathsf{hp}_{i,1} \mathsf{hp}_{i,2}^{\theta})^{s_{i,M_i}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, \delta) = H'. \end{aligned}$$

This SPHF verifies the strong pseudo-randomness property, as shown in Appendix D.2. More precisely, we have: $\mathsf{Succ}^{\texttt{c-s-ps-rand}}_{\mathcal{E}^2\mathcal{C},\mathcal{F}}(t) \leq 2 \cdot \mathsf{Succ}^{\texttt{robust}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Succ}^{\texttt{s-bind-ext}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Adv}^{\texttt{smooth}}_{\mathcal{F}}$.

5.5 Complexity and Comparisons

As summarized in Table 1, the communication complexity is linear in $m \cdot \mathfrak{K}$ (where m is the bit-length of the committed value and $\mathfrak{K}$ is the security parameter), which is much better than [ACP09] that was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m (as long as $m = \mathcal{O}(\mathfrak{K})$).

Basically, the first scheme in [FLM11] consists of a Cramer-Shoup-like encryption C of the message x, and a perfectly-sound Groth-Sahai [GS08] NIZK π that C contains x. The actual commitment is C and the opening value on x is $\delta = \pi$. The trapdoor-setup provides the Cramer-Shoup decryption key and changes the Groth-Sahai setup to the perfectly-hiding setting. The indistinguishable setups of the Groth-Sahai mixed commitments ensure the setup-indistinguishability. The extraction algorithm uses the Cramer-Shoup decryption algorithm, while the equivocation uses the simulator of the NIZK. The IND-CCA security notion for C and the computational soundness of π make it strongly-binding-extractable, the IND-CCA security notion and the zero-knowledge property of the NIZK provide the strong-simulation-indistinguishability. It is thus UC-secure. However, the verification is not robust: because of the perfectly-hiding setting of Groth-Sahai proofs, for any ciphertext C and for any message x, there exists a proof π that makes the verification of C on x. As a consequence, it is not SPHF-friendly. The second construction is in the same vein: they cannot be used in the following applications.

Table 2. Comparison with existing UC-secure PAKE schemes

	Adaptive	One-round	Communication complexity	Assumption
[ACP09][a]	yes	no	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G} + \text{OTS}^{b}$	DDH
[KV11]	no	yes	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[BBC+13]	no	yes	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
this paper	yes	yes	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH

[a] with the commitment variant of note "a" of Table 1.
[b] OTS: one-time signature (public key size and signature size) to link the flows in the PAKE protocol.

6 Password-Authenticated Key Exchange

6.1 A Generic Construction

The ideal functionality of a Password-Authenticated Key Exchange (PAKE) is depicted in Appendix A.3. It has been proposed in [CHK+05]. In Figure 7, we describe a one-round PAKE that is UC-secure against adaptive adversaries, assuming erasures. It can be built from any SPHF-friendly commitment scheme (that is $\mathtt{E}^2$, strongly-simulation-indistinguishable, and robust as described in Section 5), if the SPHF is actually a KV-SPHF [BBC+13] and the algorithms HashKG and ProjKG do not need to know the committed value π (nor the word (ℓ, C) itself), for which the SPHF verifies the strong pseudo-randomness property. We thus denote L_π the language of the pairs (ℓ, C), where C is a commitment that opens to π under the label ℓ, and L the union of all the L_π (L does not depend on π). The proof of the following theorem, with the cost of the reduction, are given in Appendix D.

Theorem 7. *The Password-Authenticated Key-Exchange described on Figure 7 is UC-secure in the presence of adaptive adversaries, assuming erasures, as soon as the commitment scheme is SPHF-friendly with a KV-SPHF.*

6.2 Concrete Instantiation

Using our commitment $\mathcal{E}^2\mathcal{C}$ introduced Section 4 together with the second SPHF described Section 5 (which satisfies the above requirements for HashKG and ProjKG), one gets a quite efficient protocol, described in Appendix E. More precisely, for m-bit passwords, each player has to send $\mathsf{hp} \in \mathbb{G}_1^{2m}$ and $C \in \mathbb{G}_1^{8m} \times \mathbb{G}_2^{m}$, which means $10m$ elements from $\mathbb{G}_1$ and m elements from $\mathbb{G}_2$. In Table 2, we compare our new scheme with some previous UC-secure PAKE.

CRS: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.

Protocol execution by P_i with π_i:

1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(L)$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, L, \perp)$ and erases any random coins used for the generation
2. P_i computes $(C_i, \delta_i) \xleftarrow{\$} \mathsf{Com}^{\ell_i}(\pi_i)$ with $\ell_i = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i stores δ_i, completely erases random coins used by Com and sends hp_i, C_i to P_j

Key computation: Upon receiving hp_j, C_j from P_j

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, L_{\pi_i}, (\ell_i, C_i), \delta_i)$ and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, L_{\pi_i}, (\ell_j, C_j))$ with $\ell_j = (\mathrm{sid}, \mathrm{ssid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H'_i \cdot H_j$ and erases everything else, except π_i.

Fig. 7. UC-Secure PAKE from an SPHF-Friendly Commitment

7 Oblivious Transfer

7.1 A Generic Construction

The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Appendix A.3. It is inspired from [CKWZ13]. In Figure 8, we describe a 3-round OT that is UC-secure against adaptive adversaries, and a 2-round variant which is UC-secure against static adversaries. They can be built from any SPHF-friendly commitment scheme, where L_t is the language of the commitments that open to t under the associated label ℓ, and from any $\mathtt{IND\text{-}CPA}$ encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with plaintext size at least $\mathfrak{K}$, and from any Pseudo-Random Generator (PRG) F with input size equal to plaintext size, and output size equal to the size of the messages in the database. Details on encryption schemes and PRGs can be found in Appendix A.2. Notice the adaptive version can be seen as a variant of the static version where the last flow is sent over a somewhat secure channel, as in [CKWZ13]; and the preflow and pk and c are used to create this somewhat secure channel.

The proof of the following theorem, with the cost of the reduction, are given in Appendix D.

CRS: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathtt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$.

Pre-flow (for adaptive security only)**:**

1. P_i generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{param})$ for $\mathcal{E}$
2. P_i stores sk, completely erase random coins used by KeyGen, and sends pk to P_i

Index query on s**:**

1. P_j chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk: $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$ (for adaptive security only; for static security: $c = \perp, R = 0$)
2. P_j computes $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$
3. P_j stores δ, completely erases S and random coins used by Com and $\mathsf{Encrypt}$, and sends C and c to P_i

Database input $(m_1, \ldots, m_k)$**:**

1. P_i decrypts $S \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$ and gets $R \leftarrow F(S)$ (for static security: $R = 0$)
2. P_i computes $\mathsf{hk}_t \xleftarrow{\$} \mathsf{HashKG}(L_t)$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, L_t, (\ell, C))$, $K_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, L_t, (\ell, C))$, and $M_t \leftarrow R \oplus K_t \oplus m_t$, for $t = 1, \ldots, k$
3. P_i erases everything except $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$ and sends $(\mathsf{hp}_t, M_t)_t$ to P_j

Data recovery:

Upon receiving $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$, P_j computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta)$ and gets $m_s \leftarrow R \oplus K_s \oplus M_s$.
Then P_j erases everything except m_s and s

Fig. 8. UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment (for Adaptive and Static Security)

Theorem 8. *The two Oblivous Transfer schemes described in Figure 8 are UC-secure in the presence of adaptive adversaries and static adversaries respectively, assuming reliable erasures and authenticated channels, as soon as the commitment scheme is SPHF-friendly.*

7.2 Concrete Instantiation and Comparison

Using our commitment $\mathcal{E}^2\mathcal{C}$ introduced Section 4 together with the first SPHF described Section 5, one gets the protocol described in Appendix E, where the number of bits of the commited value is $m = \lceil \log k \rceil$. For the statically secure version, the communication cost is, in addition to the database $\boldsymbol{m}$ that is sent in $\boldsymbol{M}$ in a masked way, 1 element of $\mathbb{Z}_p$ and k elements of $\mathbb{G}_1$ (for $\mathbf{hp}$, by using the same scalar ε for all hp_t's) for the sender, while the receiver sends $\lceil \log k \rceil$ elements of $\mathbb{G}_2$ (for $\boldsymbol{a}$) and $\lceil 8 \log k \rceil$ elements of $\mathbb{G}_1$ (for $\boldsymbol{b}$), in only two rounds. In the particular case of $k = 2$, the scalar can be avoided since the message consists of 1 bit, so our construction just requires: 2 elements from $\mathbb{G}_1$ for the sender, and 1 from $\mathbb{G}_2$ and 8 from $\mathbb{G}_1$ for the receiver, in two rounds. For the same security level (static corruptions in the CRS, with erasures), the best known solution from [CKWZ13] required to send at least 23 group elements and 7 scalars, in 4 rounds. If

adaptive security is required, our construction requires 3 additional elements in $\mathbb{G}_1$ and 1 additional round, which gives a total of 13 elements in $\mathbb{G}_1$, in 3 rounds. This is also more efficient then the best known solution from [CKWZ13], which requires 26 group elements and 7 scalars, in 4 rounds.

Acknowledgments

We thank Ralf Küsters for his comments on a preliminary version. This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet. The third author was funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *CRYPTO 2009*, *LNCS* 5677, pages 671–689. Springer, August 2009. (Pages 1, 2, 3, 9, 13, and 14.)

BBC+13. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475. Springer, 2013. Full version available on the Cryptology ePrint Archive as reports 2013/034 and 2013/341. (Pages 1, 10, 11, 14, 23, and 24.)

BCL+05. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In *CRYPTO 2005*, *LNCS* 3621, pages 361–377. Springer, August 2005. (Page 2.)

Bea96. Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *28th ACM STOC*, pages 629–638. ACM Press, May 1996. (Page 2.)

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Page 1.)

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In *TCC 2012*, *LNCS* 7194, pages 94–111. Springer, March 2012. (Page 1.)

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Pages 1 and 2.)

Can05. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. (Page 7.)

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO 2001*, *LNCS* 2139, pages 19–40. Springer, August 2001. (Pages 2, 3, 7, and 9.)

CHK+05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In *EUROCRYPT 2005*, *LNCS* 3494, pages 404–421. Springer, May 2005. (Pages 1, 14, and 19.)

CK02. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT 2002*, *LNCS* 2332, pages 337–351. Springer, April / May 2002. (Page 1.)

CKWZ13. Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2013. (Pages 1, 2, 4, 15, 16, and 19.)

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. (Pages 2, 3, 7, and 9.)

CR03. Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO 2003*, *LNCS* 2729, pages 265–281. Springer, August 2003. (Page 7.)

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*, *LNCS* 1462, pages 13–25. Springer, August 1998. (Pages 1 and 5.)

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, April / May 2002. (Pages 1 and 10.)

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Page 2.)

FLM11. Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In *ASIACRYPT 2011*, *LNCS* 7073, pages 468–485. Springer, December 2011. (Pages 3 and 13.)

GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, *LNCS* 2656, pages 524–543. Springer, May 2003. (Pages 1, 10, and 11.)

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Page 1.)

GMW91. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. (Page 1.)

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, *LNCS* 4965, pages 415–432. Springer, April 2008. (Page 13.)

Har11. Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications*. PhD thesis, New York University, 2011. (Pages 3, 5, and 9.)

HK07. Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In *CRYPTO 2007*, *LNCS* 4622, pages 111–129. Springer, August 2007. (Page 2.)

HMQ04. Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In *TCC 2004*, *LNCS* 2951, pages 58–76. Springer, February 2004. (Page 2.)

Kal05. Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT 2005*, *LNCS* 3494, pages 78–95. Springer, May 2005. (Page 1.)

Kat07. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT 2007*, *LNCS* 4515, pages 115–128. Springer, May 2007. (Page 2.)

KOY01. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001*, *LNCS* 2045, pages 475–494. Springer, May 2001. (Page 1.)

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, March 2011. (Pages 1 and 14.)

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Page 2.)

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, *LNCS* 576, pages 129–140. Springer, August 1992. (Pages 3 and 5.)

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, *LNCS* 5157, pages 554–571. Springer, August 2008. (Page 2.)

Rab81. Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981. (Page 2.)

A Notations

We first recall the classical definitions on distances of distribution, and the notions of success and advantage. We then review the basic cryptographic tools, with the corresponding security notions.

A.1 Distances, Advantage and Success

Statistical Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is also the statistical distance between X_0 and X_1:

$$\mathtt{Dist}(\mathcal{D}_0, \mathcal{D}_1) = \mathtt{Dist}(X_0, X_1) = \sum_{x \in \mathcal{S}} |\Pr[X_0 = x] - \Pr[X_1 = x]| .$$

If the statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is less than or equal to ε, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_0$ and $\mathcal{D}_1$ are 0-close, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are perfectly indistinguishable.

Success/Advantage. When one considers an experiment $\mathsf{Exp}^{\mathtt{sec}}_{\mathcal{A}}(\mathfrak{K})$ in which an adversary $\mathcal{A}$ plays a security game $\mathtt{SEC}$, we denote $\mathsf{Succ}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathtt{sec}}_{\mathcal{A}}(\mathfrak{K}) = 1]$ the success probability of this adversary. We additionally denote $\mathsf{Succ}^{\mathtt{sec}}(t) = \max_{\mathcal{A} \le t}\{\mathsf{Succ}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal success any adversary running within time t can get.

When one considers a pair of experiments $\mathsf{Exp}^{\mathtt{sec}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$, for $b = 0, 1$, in which an adversary $\mathcal{A}$ plays a security game $\mathtt{SEC}$, we denote $\mathsf{Adv}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\mathsf{Exp}^{\mathtt{sec}-0}_{\mathcal{A}}(\mathfrak{K}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathtt{sec}-1}_{\mathcal{A}}(\mathfrak{K}) = 1\right]$ the advantage of this adversary. We additionally denote $\mathsf{Adv}^{\mathtt{sec}}(t) = \max_{\mathcal{A} \le t}\{\mathsf{Adv}^{\mathtt{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal advantage any adversary running within time t can get.

Computational Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The computational distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is the best advantage an adversary can get in distinguishing X_0 from X_1: $\mathsf{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A}, \mathfrak{K}) = \Pr[\mathcal{A}(X_0) = 1] - \Pr[\mathcal{A}(X_1) = 1]$, and thus $\mathsf{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(t) = \max_{\mathcal{A} \le t}\{\mathsf{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A}, \mathfrak{K})\}$. When $\mathsf{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(t) \le \varepsilon$, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

We can note that for two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ that are ε-close, for any t and ε, $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

A.2 Formal Definitions of the Basic Primitives

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions 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 $H \xleftarrow{\$} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[H \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(H) : H(m_0) = H(m_1)\right] .$$

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 $\mathtt{param}$ of the scheme;
- $\mathsf{KeyGen}(\mathtt{param})$ generates a pair of keys, the public encryption key pk and the private decryption key sk;
- $\mathsf{Encrypt}^{\ell}(\mathsf{pk}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key pk, using the random coins r;
- $\mathsf{Decrypt}^{\ell}(\mathsf{sk}, c)$ 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 $(\mathsf{pk}, \mathsf{sk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}^{\ell}(\mathsf{sk}, \mathsf{Encrypt}^{\ell}(\mathsf{pk}, m; r)) = m.$$

– *Indistinguishability under chosen-ciphertext attacks*: this security notion IND-CCA can be formalized by the following experiments $\mathsf{Exp}^{\texttt{ind-cca-}b}_{\mathcal{A}}(\mathfrak{K})$, where the adversary $\mathcal{A}$ transfers some internal state state between the various calls FIND and GUESS, and makes use of the oracle ODecrypt:

- $\mathsf{ODecrypt}^{\ell}(c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key sk. The input queries (ℓ, c) are added to the list CTXT.

$\mathsf{Exp}^{\texttt{ind-cca-}b}_{\mathcal{A}}(\mathfrak{K})$
  $\mathtt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$
  $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{param})$
  $(\ell^*, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathtt{FIND} : \mathsf{pk})$
  $c^* \leftarrow \mathsf{Encrypt}^{\ell^*}(\mathsf{pk}, m_b)$
  $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathsf{state}, \mathtt{GUESS} : c^*)$
  if $(\ell^*, c^*) \in \mathtt{CTXT}$ **then return** 0
  else return b'

According to the previous section, these experiments implicitly define the advantages $\mathsf{Adv}^{\texttt{ind-cca}}_{\mathcal{E}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}^{\texttt{ind-cca}}_{\mathcal{E}}(t)$. One sometimes use $\mathsf{Adv}^{\texttt{ind-cca}}_{\mathcal{E}}(q_d, t)$ to bound the number of decryption queries.

Pseudo-Random Generators (PRGs). A pseudo-random generator (PRG) is a function F so that for a randomly chosen seed $S \xleftarrow{\$} \{0,1\}^{\mathfrak{K}}$ outputs a random-looking value in $\{0,1\}^{\ell}$, for some $\ell > \mathfrak{K}$.

The quality of a PRG is measured by the computational distance $\mathsf{Adv}^{\mathtt{prg}}_{F}(t)$ between the distributions of the outputs on random inputs from random values in $\{0,1\}^{\ell}$.

A.3 Ideal Functionalities

UC-Secure Oblivious Transfer. The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Figure 9. It is inspired from [CKWZ13].

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

- **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ from party P_i**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ from party P_j,** with $s \in \{1, \ldots, k\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Receive-message with the same ssid from P_j.
- **Upon receiving a message $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.
- **Upon receiving a message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m_s)$ to P_j and ignore further Received-message with the same ssid from the adversary.

Fig. 9. Ideal Functionality for 1-out-of-k Oblivious Transfer $\mathcal{F}_{(1,k)\text{-OT}}$

UC-Secure Password-Authenticated Key Exchange. We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 10). It was described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving a query (`NewSession`, sid, ssid, P_i, P_j, π) from party P_i:**
  Send (`NewSession`, sid, ssid, P_i, P_j) to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record (sid, ssid, P_j, P_i, π'), then record (sid, ssid, P_i, P_j, π) and mark this record `fresh`.
- **Upon receiving a query (`TestPwd`, sid, ssid, P_i, π') from the adversary S:**
  If there is a record of the form (P_i, P_j, π) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If $\pi \neq \pi'$, mark the record `interrupted` and reply with "wrong guess".
- **Upon receiving a query (`NewKey`, sid, ssid, P_i, sk) from the adversary S:**
  If there is a record of the form (sid, ssid, P_i, P_j, π), and this is the first `NewKey` query for P_i, then:
  - If this record is `compromised`, or either P_i or P_j is corrupted, then output (sid, ssid, sk) to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, π') with $\pi' = \pi$, and a key sk$'$ was sent to P_j, and (P_j, P_i, π) was `fresh` at the time, then output (sid, ssid, sk$'$) to P_i.
  - In any other case, pick a new random key sk$'$ of length $\mathfrak{K}$ and send (sid, ssid, sk') to P_i.

  Either way, mark the record (sid, ssid, P_i, P_j, π) as `completed`.

Fig. 10. Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the `NewKey`-query, it additionally learns the session key.

B Cramer-Shoup Encryption on Vectors

B.1 The Computational Assumption

Definition 9 (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 denote by $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple in the group $\mathbb{G}$ within time t.

B.2 Description of the Cramer Shoup Encryption on Vectors

Labeled Cramer-Shoup Encryption. The labeled Cramer-Shoup encryption scheme works in a cyclic group $\mathbb{G}$ of prime order p, with two independent generators g and h. For random scalars $x_1, x_2, y_1, y_2, z \xleftarrow{\$} \mathbb{Z}_p$, we set $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ to be the private decryption key and $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$ to be the public encryption key, where H is a random collision-resistant hash function from $\mathcal{H}$ (actually, second-preimage resistance is enough).

If $M \in G$, the Cramer-Shoup encryption is defined as $\mathsf{CS}^{\ell}_{\mathsf{pk}}(M;r) = (u = g^r, v = h^r, e = f^r \cdot M, w = (cd^{\theta})^r)$, where $\theta = H(\ell, u, v, e)$. Such a ciphertext $C = (u, v, e, w)$ is decrypted by $M = e/u^z$, after having checked the validity of the ciphertext: $w \stackrel{?}{=} u^{x_1+\theta y_1} v^{x_2+\theta y_2}$.

This encryption scheme is well-known to be IND-CCA under the DDH assumption.

Labeled Cramer-Shoup Encryption on Vectors. The above scheme can be extended to encrypt vectors of group elements $\boldsymbol{M} = (M_1, \ldots, M_m) \in \mathbb{G}^m$: $m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{M};\boldsymbol{r}) = (\mathsf{CS}^{\ell}_{\mathsf{pk}}(M_i;r_i))_i = (u_i = g^{r_i}, v_i = h^{r_i}, e_i = f^{r_i} \cdot M_i, w_i = (cd^{\theta})^{r_i})_i$, where $\theta = H(\ell, (u_i, v_i, e_i)_i)$, with indices range in $\{1, \ldots, m\}$. Such a ciphertext $C = (u_i, v_i, e_i, w_i)_i$ with label ℓ is decrypted by $M_i = e_i/u_i^z$, after having checked the validity of the ciphertext: $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$ for $i = 1, \ldots, m$.

This encryption scheme MCS is also IND-CCA under the DDH assumption. More precisely, if q_d is the number of decryption queries,

$$\mathsf{Adv}^{\mathtt{ind\text{-}cca}}_{\mathsf{MCS}}(m, q_d, t) \leq 2m \times \mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t) + \frac{m(m+1)q_d}{p}.$$

Proof. Let us be given a tuple $(g, h, u, v) \in \mathbb{G}^4$. We choose random scalars $x_1, x_2, y_1, y_2, z \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, and set $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ and $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$, where H is a random collision-resistant hash function from $\mathcal{H}$. We now consider an IND-CCA adversary on m-element vectors.

Game G_0: With the above setup, we play the IND-CCA game where the challenge ciphertext is on $\boldsymbol{M}$, that is chosen at random among the two vectors outputted by the adversary in the FIND stage.

Game G_1: In this game, instead of setting $f = g^z$, one sets $f = g^{z_1}h^{z_2}$, which essentially means that $z = z_1 + sz_2$, where $h = g^s$. Then, the decryption works as follows: $M_i = e_i/(u_i^{z_1}v_i^{z_2})$, after having checked the validity of the ciphertext: $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$ for $i = 1, \ldots, m$, relatively to the label ℓ.

One can note that the decryption algorithm yields the same result for correct ciphertexts (*i.e.*, ciphertexts such that, for $i = 1, \ldots, m$, $u_i = g^{r_i}$ and $v_i = h^{r_i}$ for some r_i): $u_i^{z_1}v_i^{z_2} = (g^{r_i})^{z_1}(h^{r_i})^{z_2} = (g^{r_i})^{z_1+sz_2} = u_i^z$.

Let us show that any adversary (even unbounded) cannot generate an incorrect ciphertext which passes the validity test $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, for all i, with non-negligible probability. Let us indeed consider such incorrect ciphertext. For some index i, $w_i = u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, but $u_i = g^{r_i}$ and $v_i = h^{r'_i}$ with $r_i \neq r'_i$. By taking the discrete logarithm in base g, and by setting $\delta_i = r'_i - r_i$, we get:

$$\begin{array}{ll} \log c = x_1 + sx_2 & \log d = y_1 + sy_2 \\ \log w_i = r_i(x_1 + \theta y_1) + sr'_i(x_2 + \theta y_2) = r_i(\log c + \theta \log d) + s\delta_i(x_2 + \theta y_2). & \end{array}$$

Since c and d do not reveal any information about x_2 and y_2, $s\delta_i(x_2 + \theta y_2)$ is unpredictable and thus the correct value for w_i is unpredicatble too. An incorrect ciphertext is declared valid with probability less than m/p, and thus the distance between the two games is bounded by mq_d/p, where q_d is the number of decryption queries.

Game G_2: In this game the challenge ciphertext is generated following the new decryption approach: $C^* = (u_i^*, v_i^*, e_i^* = u_i^{*z_1}v_i^{*z_2} \cdot M_i, w_i^* = u_i^{*x_1+\theta^* y_1} v_i^{*x_2+\theta^* y_2})_i$, where $\theta^* = H(\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, with all the tuples (g, h, u_i^*, v_i^*) being DH tuples. As already shown above, since truly DH tuples are used, this makes no difference.

Game G_3: In this game, the challenge ciphertext $C^* = (u_i^*, v_i^*, e_i^*, w_i^*)_i$, uses tuples $(u_i^*, v_i^*, e_i^*, w_i^*)$ randomly and independently chosen in $\mathbb{G}^4$.

In order to bound the distance between the two games, we use a sequence of hybrid games: in G_j, for $i \leq j$, (g, h, u_i^*, v_i^*) are DH tuples and $e_i^* = u_i^{*z_1}v_i^{*z_2} \cdot M_i$, $w_i^* = u_i^{*x_1+\theta^* y_1} v_i^{*x_2+\theta^* y_2}$, where $\theta^* = H(\ell, (u_i^*, v_i^*, e_i^*)_i)$, while for $i > j$, tuples $(u_i^*, v_i^*, e_i^*, w_i^*)$ are randomly and independently chosen in $\mathbb{G}^4$. One can note that G_0 is exactly $\boldsymbol{G}_3$, while G_m is $\boldsymbol{G}_2$.

Let us modify a little bit G_j into G'_j, in which (g, h, u, v) is a DH tuple and $u_j^* = u$, $v_j^* = v$, $e_j^* = u_j^{*z_1}v_j^{*z_2} \cdot M_j$, $w_j^* = u_j^{*x_1+\theta^* y_1} v_j^{*x_2+\theta^* y_2}$. This does not change anything. We now alter it into G''_j, where

(g, h, u, v) is a random tuple. In such a case, $e_j^* = u^{z_1} v^{z_2} \cdot M_j = u^z h^{\delta z_2} \cdot M_j$, where $\delta = r' - r$, with $u = g^r$ and $v = h^{r'}$, and thus $\delta \neq 0$ with overwhelming probability, as well as $z_2 \neq 0$. In addition, z_2 is totally unpredictable, unless some incorrect ciphertexts are decrypted.
As a conclusion, unless some incorrect ciphertexts are decrypted, e_j^* is totally unpredictable. But an incorrect ciphertext $C = (u_i, v_i, e_i, w_i)_i$ is decrypted if for some i

$$\begin{aligned} \log c &= x_1 + s x_2 \qquad \log d = y_1 + s y_2 \\ \log w_j^* &= r(x_1 + \theta^* y_1) + s r'(x_2 + \theta^* y_2) \\ \log w_i &= r_i(x_1 + \theta y_1) + s r_i'(x_2 + \theta y_2) \end{aligned}$$

whereas $\delta_i = r_i' - r_i \neq 0$. The determinant of the system is $s^2 \delta \delta_i (\theta^* - \theta)$, which is non-zero, unless $\theta^* = \theta$. But two cases only are possible:
- if $(\ell, (u_i, v_i, e_i)_i) \neq (\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, $\theta^* = \theta$ leads to a collision for H;
- if $(\ell, (u_i, v_i, e_i)_i) = (\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, then the query is not allowed.

Therefore, the determinant being non-zero, w_i is unpredictable and thus the probability that at least one incorrect ciphertext is declared valid is bounded by mq_d/p.
In addition, w_j^* is also unpredictable. This means that G_j'' is exactly G_{j-1}. but the distance between the two games is bounded by $mq_d/p + 2\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t)$, assuming no collision for H. Eventually, the distance between $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ is bounded by $m^2 q_d/p + 2m\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t)$.

This concludes the proof since in the last game, the values e_i^* are independent of the message, and thus of the bit involved in the security game: the advantage of the adversary is 0. □

We now provide a stronger security notion for encryption of vectors, which is useful for our application to $\mathtt{E}^2$-commitments.

B.3 Vector-Indistinguishability with Partial Opening, under Chosen-Ciphertext Attacks

New Security Notion. In our applications, when encrypting vectors, the adversary will get some of the random coins used for encryption. We thus define a stronger security notion:

Vector-indistinguishability with partial opening, under chosen-ciphertext attacks: this security notion VIND-PO-CCA can be formalized by the following experiments $\mathsf{Exp}^{\mathtt{vind\text{-}po\text{-}cca}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$, where the adversary $\mathcal{A}$ keeps some internal state between the various calls FIND and GUESS, and makes use of the above ODecrypt oracle. However, Encrypt* has an additional input Δ, that consists of the common values in $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, and $\perp$ at the places of distinct values. It also outputs the values $\boldsymbol{r}$ that allow to check that C^* actually encrypts a vector $\boldsymbol{M}$ that corresponds to Δ (*i.e.*, that is equal to Δ for places different than $\perp$). The exact definition of these values $\boldsymbol{r}$ depend on the actual encryption scheme.

$\mathsf{Exp}^{\mathtt{vind\text{-}po\text{-}cca}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$
$\mathtt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{param})$
$(\ell^*, \boldsymbol{M}_0, \boldsymbol{M}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{ODecrypt}'(\cdot)}(\mathtt{FIND} : \mathsf{pk})$
$\Delta = \boldsymbol{M}_0 \cap \boldsymbol{M}_1$
$(C^*, \boldsymbol{r}) \xleftarrow{\$} \mathsf{Encrypt}^{*\ell^*}(\mathsf{pk}, \Delta, \boldsymbol{M}_b)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{ODecrypt}'(\cdot)}(\mathsf{state}, \mathtt{GUESS} : C^*, \boldsymbol{r})$
if $(\ell^*, C^*) \in \mathtt{CTXT}$ **then return** 0
else return b'

This models the fact that when distinct random coins are used for each components of the vector, the random coins of the common components can be revealed, it should not help to distinguish which vector has been encrypted.

These experiments $\mathsf{Exp}^{\mathtt{vind\text{-}po\text{-}cca}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$ define the advantages $\mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathcal{E}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathcal{E}}(t)$. As above, we will use $\mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathcal{E}}(m, q_d, \gamma, t)$ to make precise the length m of the vectors, and to bound by q_d the number of decryption queries and by γ the number of distinct values in the pairs of vectors.

Labeled Cramer-Shoup Encryption on Vectors. For the Cramer-Shoup encryption on vectors MCS, the values $\boldsymbol{r}$ output by $\mathsf{Encrypt}^*$ are the random coins r_i corresponding to to the common components of $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$ (*i.e.*, for i such that $M_{0,i} = M_{1,i} = \Delta_i$). These values are sufficient to check that C^* actually encrypts a vector corresponding to Δ.

We can prove that MCS is $\texttt{VIND-PO-CCA}$ using a slight variant of the $\texttt{IND-CCA}$ proofs given in Section B.2. More precisely, we use the same games except that for i such that $M_{0,i} = M_{1,i}$, in Games $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$, we compute u_i^* and v_i^* as in Game $\boldsymbol{G}_1$: $u_i^* = g^{r_i}$ and $v_i^* = h^{r_i}$ for a random scalar r_i. The hybrid technique to prove the indistinguishability of $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ uses γ steps only (instead of m steps), with γ the number of distinct components. Finally, we get:

$$\mathsf{Adv}^{\texttt{vind-po-cca}}_{\mathsf{MCS}}(m, q_d, \gamma, t) \leq 2\gamma \times \mathsf{Adv}^{\texttt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\texttt{coll}}_{\mathcal{H}}(t) + \frac{m(\gamma+1)q_d}{p}.$$

C Useful SPHFs

In this appendix, we show three constructions of SPHF for $\mathcal{E}^2\mathcal{C}$ and summarize their costs. We use the framework from [BBC+13].

C.1 Construction of the SPHFs

The language we are interested in is the language of valid commitments $C = (\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{G}_2^m \times \mathbb{G}_1^{8m}$ under label ℓ of some fixed vector $\boldsymbol{M}$, where the witness is the decommitment information $\delta = \boldsymbol{s} = (s_{i,M_i})_i \in \mathbb{Z}_p^m$. More precisely, a word C is in the language if and only if there exists δ and $\boldsymbol{d} = (d_{i,M_i})_i \in \mathbb{G}_1^m$ such that, for all $i = 1, \ldots, m$:

$$b_{i,M_i} = \mathsf{CS}^{\ell'}_{\mathsf{pk}}(d_{i,M_i}, \theta; s_{i,M_i}) \quad \text{and} \quad e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2),$$

with $\ell' = (\ell, \boldsymbol{a})$ and $\theta = H(\ell', (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$. If we set $b_{i,j} = (u_{i,j}, v_{i,j}, e_{i,j}, w_{i,j})_{i,j}$, we can write the previous conjunction as: for all $i = 1, \ldots, m$,

$$(u_{i,M_i}, v_{i,M_i}, e_{i,M_i}, w_{i,M_i}) = (g_1^{s_{i,M_i}}, h_1^{s_{i,M_i}}, f_1^{s_{i,M_i}} \cdot d_{i,M_i}, (cd^\theta)^{s_{i,M_i}}) \quad \text{and} \quad e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2).$$

Since e is non-degenerated, we finally remark that we can eliminate the need of d_{i,M_i}, by lifting everything in $\mathbb{G}_T$, and checking that, first, the ciphertexts are all valid and, second, the plaintexts satisfy the appropriate relations:

$$\begin{aligned}(e(u_{i,M_i}, g_2), e(v_{i,M_i}, g_2), e(w_{i,M_i}, g_2)) &= (e(g_1^{s_{i,M_i}}, g_2), e(h_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2)) \\ e(e_{i,M_i}, g_2) &= e(f_1^{s_{i,M_i}}, g_2) \cdot e(g_1, a_i/T^{M_i}).\end{aligned}$$

From these expressions we can derive three SPHFs.

KV-SPHF. A KV-SPHF is a SPHF for which hp does not depend on C and the smoothness holds even if the adversary can see hp before choosing C (see [BBC+13] for a precise definition). Let us first show how to construct a KV-SPHF checking the previous condition for only a fixed index i. For this purpose, we use the following matrices (for the framework of [BBC+13]:

$$\begin{aligned}\Gamma &= \begin{pmatrix} g_1 & 0 & h_1 & f_1 & c \\ 0 & g_1 & 0 & 0 & d \end{pmatrix} \qquad (2)\\ \boldsymbol{\lambda}_i &= (g_2^{s_{i,M_i}}, g_2^{s_{i,M_i}\theta}) \\ \boldsymbol{\lambda}_i \cdot \Gamma &= (e(g_1^{s_{i,M_i}}, g_2), e(g_1^{s_{i,M_i}\theta}, g_2), e(h_1^{s_{i,M_i}}, g_2), e(f_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2)) \\ \Theta_i(C) &= (e(u_{i,M_i}, g_2), e(u_{i,M_i}^\theta, g_2), e(v_{i,M_i}, g_2), e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}), e(w_{i,M_i}, g_2))\end{aligned}$$

With $\mathsf{hk} = (\eta_1, \eta_2, \alpha, \beta, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, we get $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} h_1^{\alpha} f_1^{\beta} c^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu}) \in \mathbb{G}_1^2$.

Eventually, to check that C actually commits to $\boldsymbol{M} = (M_i)_i$, we can define a KV-SPHF, as follows:

$$\begin{aligned}
\mathsf{hk} = (\eta_{i,1}, \eta_{i,2}, \alpha_i, \beta_i, \mu_i)_i \xleftarrow{\$} \mathbb{Z}_p^{5m} \quad \mathsf{hp} = (\mathsf{hp}_{i,1} = g_1^{\eta_{i,1}} h_1^{\alpha_i} f_1^{\beta_i} c^{\mu_i}, \mathsf{hp}_{i,2} = g_1^{\eta_{i,2}} d^{\mu_i})_i \in \mathbb{G}_1^{2m},\\
H = \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \stackrel{\text{def}}{=} \prod_i \Big(e(u_{i,M_i}^{(\eta_{i,1}+\theta\eta_{i,2})} \cdot v_{i,M_i}^{\alpha_i}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta_i} \cdot e(w_{i,M_i}^{\mu_i}, g_2) \Big)\\
= e(\textstyle\prod_i (\mathsf{hp}_{i,1} \mathsf{hp}_{i,2}^{\theta})^{s_{i,M_i}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, (s_{i,M_i})_i) = H'.
\end{aligned}$$

Since $\delta = (s_{i,M_i})_i$, this will allow the one-round PAKE, described in Section 6.

CS-SPHF. When C is sent before hp, we can use a CS-SPHF instead of a KV-SPHF (for which the smoothness holds only when the adversary cannot see hp before choosing C). Here is a more efficient CS-SPHF, with a common projection key for all the components, but an additional random ε[1]:

$$\begin{aligned}
\Gamma = \begin{pmatrix} g_1 & 0 & h_1 & f_1 & c \\ 0 & g_1 & 0 & 0 & d \end{pmatrix} \qquad \boldsymbol{\lambda} = \Big(\textstyle\prod_i g_2^{s_{i,M_i}\varepsilon^{i-1}}, g_2^{s_{i,M_i}\theta\varepsilon^{i-1}} \Big)\\
\boldsymbol{\lambda} \cdot \Gamma = \Big(e(\textstyle\prod_i g_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2), \quad e(\textstyle\prod_i g_1^{s_{i,M_i}\theta\varepsilon^{i-1}}, g_2), \quad e(\textstyle\prod_i h_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2),\\
e(\textstyle\prod_i f_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2), \quad e(\textstyle\prod_i (cd^{\theta})^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \Big)\\
\Theta(C) = \Big(\textstyle\prod_i e(u_{i,M_i}^{\varepsilon^{i-1}}, g_2), \quad e(\textstyle\prod_i u_{i,M_i}^{\theta\varepsilon^{i-1}}, g_2), \quad e(\textstyle\prod_i v_{i,M_i}^{\varepsilon^{i-1}}, g_2),\\
\textstyle\prod_i (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\varepsilon^{i-1}}, \quad e(\textstyle\prod_i w_{i,M_i}^{\varepsilon^{i-1}}, g_2) \Big),
\end{aligned}$$

which leads to

$$\begin{aligned}
\mathsf{hk} = (\eta_1, \eta_2, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^6 \quad \mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^{\eta_1} h_1^{\alpha} f_1^{\beta} c^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu}) \in \mathbb{Z}_p \times \mathbb{G}_1^2,\\
H = \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \stackrel{\text{def}}{=} \prod_i \Big(e(u_{i,M_i}^{(\eta_1+\theta\eta_2)} \cdot v_{i,M_i}^{\alpha}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta} \cdot e(w_{i,M_i}^{\mu}, g_2) \Big)^{\varepsilon^{i-1}}\\
= e(\textstyle\prod_i (\mathsf{hp}_1 \mathsf{hp}_2^{\theta})^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, (s_{i,M_i})_i) = H'.
\end{aligned}$$

More precisely, following the framework from [BBC+13], since hp is not known at the time C is generated, to prove that the resulting CS-SPHF is smooth, we just need to prove that for any invalid C (not in the language), the probability that $\Theta(C)$ is not a *linear* combination of the rows of $\Gamma(C)$ is overwhelming, over the random choice of ε. Indeed, if $\Theta(C)$ is independent of rows of $\Gamma(C)$, H is completely unpredictable even given hp.

Let us indeed consider an invalid commitment C, and let us write $b_{i,M_i} = (u_{i,M_i}, v_{i,M_i}, e_{i,M_i}, w_{i,M_i})_i = (g_1^{s_i}, h_1^{t_i}, f_1^{s'_i} \cdot d_{i,j}, (cd^{\theta})^{s''_i})_i$, with d_{i,M_i} such that $e(d_{i,M_i}, g_2) = e(g_1, a_i/T^{M_i})$, which is always possible if we suppose g_1, h_1, f_1 and cd^{θ} are all generators[2]. Then, since C is invalid, there exists some i^*, such that either $t_{i^*} \neq s_{i^*}$ or $s'_{i^*} \neq s_{i^*}$ or $s''_{i^*} \neq s_{i^*}$. Let us suppose $t_{i^*} \neq s_{i^*}$. The other cases are similar. Then we remark that for $\Theta(C)$ to be linearly dependent of rows of Γ, it is necessary that $e(\prod_i v_{i,M_i}^{\varepsilon^{i-1}}, g_2) = e(h_1^{s_i \sum_i \varepsilon^{i-1}}, g_2)$, since the first coefficient of the linear combination is necessary $\lambda_1 = g_2^{\sum_i s_i\varepsilon^{i-1}} = \prod_i g_2^{s_i\varepsilon^{i-1}}$. This implies that: $\sum_i t_i\varepsilon^{i-1} = \sum_i s_i\varepsilon^{i-1}$, by looking at the discrete logarithm in base $e(g_1, g_2)$. In other words ε has to be a root of the m-degree polynomial $\sum_i (t_i - s_i)X^i$, which is not null because $t_{i^*} \neq s_{i^*}$. This polynomial has at most m roots, and so at most m distinct ε (among the p possible ε in $\mathbb{Z}_p$) lead to a $\Theta(C)$ linearly dependent of rows of Γ. Finally, we conclude that, with probablity at least $1 - m/p$, $\Theta(C)$ is independent of rows of Γ. This proves the smoothness of the SPHF.

[1] Actually, it is possible to choose $\varepsilon \xleftarrow{\$} \{0,1\}^{\mathfrak{K}}$.

[2] It is easy to extend to the case where some of them are not generators (*i.e.*, are equal to 1).

GL-SPHF. When C is sent in advance, and thus known when generating hp, as in the Oblivious Transfer protocol described in Section 7, we can use a more efficient GL-SPHF instead of a CS-SPHF:

$$\Gamma(C) = \left(g_1\ h_1\ f_1\ (cd^\theta)\right) \qquad \boldsymbol{\lambda}_i \cdot \Gamma(C) = (e(g_1^{s_{i,M_i}}, g_2), e(h_1^{s_{i,M_i}}, g_2), e(f_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2))$$
$$\boldsymbol{\lambda}_i = g_2^{s_{i,M_i}} \qquad \Theta_i(C) = (e(u_{i,M_i}, g_2), e(v_{i,M_i}, g_2), e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}), e(w_{i,M_i}, g_2))$$

and:

$$\mathsf{hk} = (\eta, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^5 \quad \mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^\eta h_1^\alpha f_1^\beta (cd^\theta)^\mu) \in \mathbb{Z}_p \times \mathbb{G}_1,$$
$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^\eta \cdot v_{i,M_i}^\alpha, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^\beta \cdot e(w_{i,M_i}^\mu, g_2)\right)^{\varepsilon^{i-1}}$$
$$= e(\prod_i \mathsf{hp}_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, (s_{i,M_i})_i) = H'.$$

C.2 Complexity

In Table 3, we summarize the cost of the various SPHFs. We remark that in the CS-SPHF and GL-SPHF, if $m = 1$, we do not need ε. That is why this case is handled separately. In all cases, the hash value consists of 1 group element in $\mathbb{G}_T$. In practice we use an entropy extractor on this hash value.

Table 3. Cost of the three SPHFs for $\mathcal{E}^2\mathcal{C}$

	hk	hp
KV-SPHF	$5m \times \mathbb{Z}_p$	$2m \times \mathbb{G}_1$
CS-SPHF (for $m = 1$)	$5 \times \mathbb{Z}_p$	$2 \times \mathbb{G}_1$
CS-SPHF (for $m \geq 2$)	$6 \times \mathbb{Z}_p$	$2 \times \mathbb{G}_1 + 1 \times \mathbb{Z}_p$
GL-SPHF (for $m = 1$)	$4 \times \mathbb{Z}_p$	$1 \times \mathbb{G}_1$
GL-SPHF (for $m \geq 2$)	$5 \times \mathbb{Z}_p$	$1 \times \mathbb{G}_1 + 1 \times \mathbb{Z}_p$

D Security Proofs

Each flow in the concrete protocols should include the tuple $(\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, but we omit it for the sake of simplicity. This tuple is needed for the queries the simulator to asks to the ideal functionality (in the description of the ideal games at the end of each sequence of games below).

D.1 Proof of Theorem 4 (UC-Secure Commitment from a Labeled $\mathtt{E}^2$-Commitment)

We first prove this theorem in the case the labeled $\mathtt{E}^2$-commitment scheme $\mathcal{C}$ is additionally strongly-binding-extractable, and explain the difference with strong-simulation-indistinguishability afterwards.

With Strong-Binding-Extractability. We thus exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

Essentially, one first makes the setup algorithm additionally output the trapdoor (setup-indistinguishability); one can then replace all the commitment queries by simulated (fake) commitments (simulation-indistinguishability). Eventually, when simulating the receiver, the simulator extracts the committed value x (using ExtCom), which should be the same as the one that will be open later (strong-binding-extractability). One can then split the simulation, to open it when required only with the appropriate information: the committed value sent be the environment is not required anymore. More details follow:

Game G_0: This is the real game.

Game G_1: In this game, the simulator generates correctly every flows from the honest players, as they would do themselves, knowing the inputs x sent by the environment to the senders. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game G_2: In this game, we just replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\lambda})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability*. Corruptions are handled the same way.

Game G_3: We first deal with honest senders: we replace all the commitments $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ in Step 1. of the commit phase of honest players by simulated commitments $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, which means $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$. We then store (ℓ, x, C) in Λ.

With an hybrid proof, applying the $\mathsf{Exp}^{\mathtt{sim\text{-}ind}}$ security game for each step, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games.

In case of corruption of the sender, one learns the already known value x.

Game G_4: We now deal with honest receivers: when receiving a fresh commitment C from the adversary or a replay from another session (and thus under a different label), the simulator extracts the committed value x. If the adversary later opens to a different value at the decommit phase, the simulator rejects it. If it was accepted in the previous game, then one breaks the strong-binding-extractability.

More generally, the trapdoor correctness ensures that valid decommitments (accepted in the previous game) will be accepted in this game too, which makes almost no difference between the two games, but with probability bounded by $\mathsf{Succ}^{\mathtt{s\text{-}bind\text{-}ext}}_{\mathcal{C}}(t)$. Note that in the experiment $\mathsf{Exp}^{\mathtt{s\text{-}bind\text{-}ext}}$, the adversary has access to both the fake commitment oracle (SCom) and the extraction oracle (ExtCom), which are indeed required here.

Game G_5: We do not use anymore the knowledge of x: the simulator knows the trapdoor τ and generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the commit phase of honest players. When receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game except that Λ is first filled with $(\ell, \perp, C)$ at the commit time and then updated to (ℓ, x, C) at the opening time. In case of corruption of the sender, one learns the committed value that is thereafter used at the decommit phase for x.

Game G_6: We can now make use of the functionality, which leads to the following simulator:

- when receiving a commitment C from the adversary, and thus either freshly generated by the adversary or a replay of a commitment C generated by the simulator in another session (with a different label), the simulator extracts the committed value x, and uses it to send a `Commit` message to the ideal functionality. A dummy value is used in case of bad extraction;
- when receiving a `Receipt`-message, which means that an honest player has committed a value, the simulator generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the commit phase of the honest player;
- when receiving (x, δ), if the verification succeeds, the simulator asks for a `Reveal` query to the ideal functionality;
- when receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment.

Any corruption just reveals x earlier, which allows a correct simulation of the opening.

With Strong-Simulation-Indistinguishability. In the other case, where the labeled $\mathtt{E}^2$-commitment is additionally strongly-simulation-indistinguishable, we can just first simulate the receiver before modifying the simulation of the sender with fake commitments: one will first apply extractability and then strong-simulation-indistinguishability. This concretely means that we swap $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$. This leads to the same simulator in $\boldsymbol{G}_6$.

Cost of the Reductions. More precisely, for any environment, its advantage in distinguishing the ideal world (with the ideal functionality from Figure 4) and the real world (with the commitment scheme $\mathcal{C}$) is

bounded by both $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t)+q_s\cdot\mathsf{Adv}_{\mathcal{C}}^{\texttt{sim-ind}}(t)+\mathsf{Succ}_{\mathcal{C}}^{\texttt{s-bind-ext}}(t)$ and $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t)+q_s\cdot\mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t)+\mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$, where q_s is the number of concurrent sessions and t its running time.

D.2 Proofs for Sections 4.3 and 5.4 (Security Properties of our $\mathcal{E}^2\mathcal{C}$ Commitment)

Setup-indistinguishability. this is trivially satisfied since the two setup algorithms are exactly the same but just output the trapdoor or not, and thus $\mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\texttt{setup-ind}}(t)=0$ for any t.

(t,ε)-strong-simulation-indistinguishability. Let us build a sequence of games from the security experiment with $b=1$ to the experiment with $b=0$. We stress that SCom does not only output $C=(\boldsymbol{a},\boldsymbol{b})$, but also $\delta=((d_{i,M_i},s_{i,M_i})_i)$, where the $s_{i,j}$'s are the random coins in the multi-Cramer-Shoup encryption.

1. We first start with the real game with $b=1$ (use of SCom for the challenge commitment), with all the trapdoors to emulate the oracles;
2. the simulator now knows the equivocation trapdoor to emulate the SCom-oracle, but has just access to the decryption oracle to emulate the ExtCom-oracle;
3. for the challenge oracle on $x=(x_i)_i$, the simulator uses $r_{i,1-x_i}=0$, which leads to the plaintexts $d_{i,1-x_i}=1$ that are thereafter encrypted under the Cramer-Shoup encryption scheme. Applying the `VIND-PO-CCA` security of the MCS encryption scheme, in which the m components of the vector that correspond to the committed vector x are the same in the two $2m$-long vectors, one can note that the bias is upper-bounded by $\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{vind-po-cca}}(2m,q_d,m,t)$, where q_d the number of extraction queries. The two vectors submitted to the encryption oracle $\mathsf{Encrypt}^*$ in the security game `VIND-PO-CCA` are $(d_{1,0},d_{1,1},\ldots,d_{m,0},d_{m,1})$, where the d_{i,x_i}'s keep the same in the two games, but the $d_{i,1-x_i}$'s are all replaced by 1 in the second game. Then, the $\mathsf{Encrypt}^*$ oracle additionally outputs the s_{i,x_i}'s (that correspond to the common components), which allows to output δ.
4. giving back all the trapdoors to the simulator, we are in the real game with $b=0$ (use of Com for the challenge commitment).

In conclusion, one thus gets $\mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\texttt{s-sim-ind}}(t)\leq\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{vind-po-cca}}(2m,q_d,m,t)$.

Strong binding extractability. Let us build a sequence of games from the security experiment to an attack to the DDH in $\mathbb{G}_2$.

1. we first start with the real game, with all the trapdoors to emulate the oracles;
2. the simulator replaces all the SCom-oracle queries by Com-oracle queries. With an hybrid proof, where we replace sequentially the SCom emulations by Com emulations, as above, one introduces a bias upperbounded by $q_c\cdot\mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\texttt{s-sim-ind}}(t)$, and thus $q_c\cdot\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{vind-po-cca}}(2m,q_d,m,t)$, where q_c is the number of SCom-queries and q_d the number of extract queries;
3. the simulator does not need any more the equivocation trapdoor, but can still extract the correct d_{i,x_j}, by decrypting the Cramer-Shoup ciphertexts, to open the commitment with $e(g_1,a_i/T^{x_i})=e(d_{i,x_i},g_2)$. When the adversary breaks the strong-binding-extractability, it provides C with a valid opening $(\boldsymbol{M},\delta)$, whereas C extracts to $\boldsymbol{M'}\neq\boldsymbol{M}$ (possibly $\perp$).
   Since opening/verification in one way is possible $\boldsymbol{M}$, this means that the Cramer-Shoup decryption gives at least one valid opening for each a_i. But because of the different extraction output $\boldsymbol{M'}$, extraction technique is ambiguous on C: for an index i, it can provide two different opening values for a_i, which breaks the DDH assumption in $\mathbb{G}_2$.

In conclusion, one thus gets $\mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\texttt{s-bind-ext}}(t)\leq q_c\cdot\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{vind-po-cca}}(2m,q_d,m,t)+\mathsf{Adv}_{\mathbb{G}_2}^{\texttt{ddh}}(t)$, where q_c is the number of SCom-queries and q_d the number of extract queries.

Robustness. In the above proof of strong-binding-extractability, as soon as different opening values exist, by decrypting the Cramer-Shoup ciphertexts, one breaks the DDH assumption in $\mathbb{G}_2$: $\mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\texttt{robust}}(t)\leq q_c\cdot\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{ind-cca}}(t)+\mathsf{Adv}_{\mathbb{G}_2}^{\texttt{ddh}}(t)$, where q_c is the number of SCom-queries.

Strong pseudo-randomness. For the sake of simplicity, we write $x=\boldsymbol{M}$ and $x'=\boldsymbol{M'}$. We also write: $C=(\boldsymbol{a},\boldsymbol{b})$ and $C'=(\boldsymbol{a'},\boldsymbol{b'})$. To prove the strong pseudo-randomness, we use the following sequence of games:

Game G_0: This game is the experiment $\mathsf{Exp}^{\texttt{c-s-ps-rand-0}}_{\mathcal{A}}$.

Game G_1: In this game, before computing H', we compute $\boldsymbol{M'} \leftarrow \mathsf{ExtCom}^{\ell'}(\tau, C')$ and we abort if for some i, the decryptions of $b'_{i,0}$ and $b'_{i,1}$ give valid opening values of a'_i for 0 and 1 respectively. In other words, if C' is not perfectly binding, we abort.[3]

This game is indistinguishable from the previous one, using the proof of robustness.

Game G_2: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H' by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, the fact that $\boldsymbol{M'} \neq \boldsymbol{M}$ and C' is perfectly binding (otherwise, we would have aborted), so that $(\ell', C') \notin L_{\boldsymbol{M}}$, and thanks to the fact that H could have been computed as follows: $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$ and $H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, L_{\boldsymbol{M}}, (\ell, C), \delta)$.

Game G_3: In this game, we replace $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ by $C \xleftarrow{\$} \mathsf{Com}^{\ell}(\boldsymbol{M''})$ for some arbitrary $\boldsymbol{M''} \neq \boldsymbol{M}$. This game is indistinguishable thanks to strong simulation indistinguishability (since eqk is not used, SimCom could have been replaced by SCom with a $\boldsymbol{M''}$ as message).

Game G_4: In this game, when $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that C is a real commitment of $\boldsymbol{M''} \neq \boldsymbol{M}$ and so that $(\ell, C) \notin L_{\boldsymbol{M}}$.

Notice that we could not have done this if $\boldsymbol{M'} = \boldsymbol{M}$, since, in this case, we still need to use hk to compute the hash value H' of C'. We are handling this (tricky) case in the following game.

Game G_5: In this game, we replace H by a random value, in the case $\boldsymbol{M'} = \boldsymbol{M}$. So now H will be completely random, in all cases (since it was already the case when $\boldsymbol{M'} \neq \boldsymbol{M}$).

Let $\theta = H(\ell, (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$ and $\theta' = H(\ell', (u'_{i,j}, v'_{i,j}, e'_{i,j})_{i,j})$. Finally, we write $s'_{i,j} = \log u'_{i,j}$ for all i, j, log being the discrete logarithm in base g_1. There are two cases:

1. for all i, $v'_{i,M_i} = h_1^{s'_{i,M_i}}$. In this case, since C' extracts to $\boldsymbol{M}$, this means that

$$w'_{i,M_i} = u'^{x_1+\theta' y_1}_{i,M_i} \cdot v'^{x_2+\theta' y_2}_{i,M_i},$$

   and so from the definition of c and d, we have that:

$$w'_{i,M_i} = (c \cdot d^{\theta'})^{s'_{i,M_i}}.$$

   This means that $(\ell', C') \in L_{\boldsymbol{M}}$, and its hash value H' could be computed knowing only hp and $(s'_{i,M_i})_i$. Therefore, the hash value H of C looks random by smoothness.

2. for some i, $v'_{i,M_i} \neq h_1^{s'_{i,M_i}}$. Then since $v_{i,M_i} = h_1^{s_{i,M_i}}$, the rows of the matrix Γ in Equation 2 (page 23) and the two vectors $\Theta_i(C)$ and $\Theta_i(C')$ are linearly independent. Then, even given access to the hash value H' of C' and the projection key hp, the hash value H of C looks perfectly random.

The following games are just undoing the modifications we have done, but keeping H picked at random

Game G_6: In this game, we now compute C as originally using SimCom. This game is indistinguishable thanks to strong simulation indistinguishability.

Game G_7: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we compute H' as originally (as the hash value of C').

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF.

Game G_8: In this game, we do not extract $\boldsymbol{M'}$ from C' nor abort when C' is not perfectly binding. Thanks to the robustness, this game is indistinguishable from the previous one.

We remark that this game is exactly the experiment $\mathsf{Exp}^{\texttt{c-s-ps-rand-1}}_{\mathcal{A}}$.

Finally, one thus gets $\mathsf{Succ}^{\texttt{c-s-ps-rand}}_{\mathcal{E}^2\mathcal{C},\mathcal{F}}(t) \leq 2 \cdot \mathsf{Succ}^{\texttt{robust}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Succ}^{\texttt{s-bind-ext}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Adv}^{\texttt{smooth}}_{\mathcal{F}}$.

[3] Actually, we may abort more often than that, but at least, if the commitment C' is honestly generated, we do not abort.

D.3 Proof of Theorem 7 (UC-Secure PAKE from an SPHF-Friendly Commitment)

To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}_{\text{pwKE}}$ and the adversary $\mathcal{A}$.

For the sake of simplicity, since the protocol is fully symmetric in P_i and P_j, we describe the simulation for player P_i in order to simplify the notations.

We say that a flow is *oracle-generated* if the pair (hp, C) was sent by an honest player (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game G_0: This is the **real game.**

Game G_1: First, in this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs π_i and π_j sent by the environment to the players. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.
In the following, Step 1. is always generated honestly by the simulator, since the hashing and projection keys do not depend on any private value.

Game G_2: We now replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability*. Corruptions are handled the same way.

Game G_3: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have distinct passwords**. In this case, $\mathcal{S}$ has received the password π_j of P_j at the corruption time of P_j (π_j was anyway already known), and knows the corresponding opening data δ_j, generated with the commitment by the Com-call. If this password is the same, it does not change anything. If the passwords are distinct, then $\mathcal{S}$ computes H_i' as before, but chooses H_j at random: this means that we replace $\mathsf{Hash}(\mathsf{hk}_i, L_{\pi_i}, (\ell_j, C_j))$ by a random value, while C_j has been simulated by Com with an opening value δ_j for $\pi_j \neq \pi_i$.
With an hybrid proof, applying the $\mathsf{Exp}^{\mathtt{c\text{-}smooth}}$ security game, with $x = \pi_i$ and $x' = \pi_j$ (since C_j is generated by Com on π_j, it thus extracts on $x' = \pi_j$), one proves this game is indistinguishable from the former one.

Game G_4: We conclude for this case: if the passwords are distinct, P_i chooses a random key.
Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game G_5: In this game, we simulate the **commitments sent by the honest players** using the trapdoors generated by the setup algorithm $\mathsf{SetupComT}$. More precisely, we replace the commitment $(C_i, \delta_i) \xleftarrow{\$} \mathsf{Com}^{\ell_i}(\pi_i)$ sent by an honest P_i with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$ in Step 2. by a simulated commitment $(C_i, \delta_i) \xleftarrow{\$} \mathsf{SCom}^{\ell_i}(\tau, \pi_i)$, which means $(C_i, \mathsf{eqk}_i) \xleftarrow{\$} \mathsf{SimCom}^{\ell_i}(\tau)$ and $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$. We then store $(\ell_i, \pi_i, C_i, \delta_i)$ in Λ.
With an hybrid proof, applying the $\mathsf{Exp}^{\mathtt{sim\text{-}ind}}$ security game for each player, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games.
In case of corruption of the honest player P_i, one learns the already known value π_i. If the corruption occurs before the erasures, we are able to provide the adversary with coherent values (δ_i has been computed using the correct value π_i). If the corruption occurs in the end, we are able to give the adversary the (honestly computed) session key. Unless we precise it, all the corruptions are dealt with in the same way in the following games.

Game G_6: In this game, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have identical passwords.** When P_i receives an oracle-generated flow from P_j, the simulator checks whether the two passwords sent by the environment for P_i and P_j are identical. If so, $\mathcal{S}$ computes both hash values using Hash and not $\mathsf{ProjHash}$. More precisely, it computes $H_i' = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_j}, (\ell_i, C_i))$ (with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$). If the passwords are distinct, it does not change anything. Recall that it is able to do so since it generated the hashing keys on their behalf.

Thanks to the correctness of the SPHF, this game is indistinguishable from the former one.

Game $\boldsymbol{G}_7$: Still in this case, we replace H'_i (and H_i if P_j received the oracle-generated flow generated flow sent by P_i) by a random value.

To prove this game is indistinguishable from the previous one, we consider two cases:

- P_j received the oracle-generated flow generated by P_i. In this case, hk_j is only used to compute $H_i = H'_i$, and since δ_i is no more used, we can apply the pseudo-randomness game on C_i to prove that $H_i = H'_i$ is indistinguishable from random;
- P_j received a non-oracle-generated flow (hp'_i, C'_i). In this case hk_j is only used to compute $H'_i = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_i}, (\ell_i, C_i))$ and $H_i = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_i}, (\ell_i, C'_i))$. In this case, we can apply the strong pseudo-randomness game to prove that H'_i still looks random.

Game $\boldsymbol{G}_8$: We conclude for this case: $\mathcal{S}$ sends a random key to P_i.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_9$: In the next two games, we deal with the case where P_i **receives a non-oracle-generated flow** (hp_j, C_j)**.** Since this pair is fresh, either C_j is new or hp_j (and thus the label) is new. In both cases, $\mathcal{S}$ can extract the committed value π'_j on behalf of P_j.

If this password is the same than that of P_i (which the simulator can easily check, still having access to the private values sent by the environment), $\mathcal{S}$ still computes both H_j and H'_i as before.

Otherwise (or if the extraction fails), the $\mathcal{S}$ computes H'_i as before, but chooses H_j at random:

Under the smoothness, with an hybrid proof, applying the $\mathsf{Exp}^{\texttt{c-smooth}}$ security game for each such hash value, one can show the indistinguishability of the two games.

Game $\boldsymbol{G}_{10}$: Finally, when P_i receives a non-oracle-generated flow (hp_j, C_j) that extracts to a different password than that of P_i (or for which extraction fails), then $\mathcal{S}$ sets the session key of P_i as random.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_{11}$: We do not use anymore the knowledge of π_i when simulating an honest player P_i. The simulator generates $(C_i, \mathsf{eqk}_i) \xleftarrow{\$} \mathsf{SimCom}^{\ell_i}(\tau)$, with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$, to send C_i. It then stores $(\ell_i, \perp, C_i, \mathsf{eqk}_i)$ in Λ. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game since δ_i is never revealed. Λ is first filled with $(\ell_i, \perp, C_i, \mathsf{eqk}_i)$, it can be updated with correct values in case of corruption of P_i. Indeed, in case of corruption, $\mathcal{S}$ recovers the password π_i and computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$, which it is able to give to the adversary.

The private values of P_i are thus not used anymore in Step 1. and Step 2. The simulator only needs them to choose how to set the session key of the players. In the ideal game, this will be replaced by a `NewKey`-query that will automatically deal with equality or difference of the passwords, or `TestPwd`-query for non-oracle-generated-flows.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_{12}$: This is the ideal game. Now, the simulator does not know the private values of the honest players anymore, but can make use of the ideal functionality. We showed in Game $\boldsymbol{G}_{11}$ that the knowledge of the private values is not needed anymore by the simulator, provided it can ask queries to the ideal functionality:

Initialization: When initialized with security parameter $\mathfrak{K}$, the simulator first runs the commitment setup algorithm $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$, and initializes the real-world adversary $\mathcal{A}$, giving it ρ as common reference string.

Session Initialization: When receiving a message $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ from $\mathcal{F}_{pwKE}$, $\mathcal{S}$ executes the protocol on behalf of P_i as follows:

1. $\mathcal{S}$ generates honestly $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(L)$ and $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, L, \perp)$;
2. $\mathcal{S}$ computes $(C_i, \mathsf{eqk}_i) \xleftarrow{\$} \mathsf{SimCom}^{\ell_i}(\tau)$ with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$;
3. $\mathcal{S}$ sends $(\mathrm{sid}, P_i, P_j, \mathsf{hp}_i, C_i)$ to P_j.

If P_i gets corrupted, $\mathcal{S}$ recovers the password π_i and computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}, C_i, \pi_i)$, which it is able to give to the adversary.

Key Computation: When receiving a flow (hp_j, C_j):

- if the flow (C_j, hp_j) is non-oracle-generated, $\mathcal{S}$ extracts the password π'_j (or set it as a dummy value in case of failure of extaction). $\mathcal{S}$ then asked for a `TestPwd`-query to the functionality to check whether π'_j is the password of P_i. If this password is correct, $\mathcal{S}$ sets $\pi_i = \pi'_j$, computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$, as well as H_j and H'_i, and then sk_i, that is passed to the `NewKey`-query (`compromised` case). If the password is incorrect, $\mathcal{S}$ asks the `NewKey`-query with a random key (`interrupted` case).
- if the flow (C_j, hp_j) is oracle-generated but the associated P_j has been corrupted, then $\mathcal{S}$ has recovered its password π_j and δ_j. It can thus compute sk_j, that is passed to the `NewKey`-query (`corrupted` case).
- if the flow (C_j, hp_j) is oracle-generated and the associated P_j is still uncorrupted, $\mathcal{S}$ asks the `NewKey`-query with a random key (normal case).

One can remark that the `NewKey`-queries will send back the same kinds of session keys to the environment as in Game $\boldsymbol{G}_{11}$: if a player is corrupted, the really computed key is sent back, in case of impersonation attempt, the `TestPwd`-query will address the appropriate situation (correct or incorrect guess), and if the two players are honest, the `NewKey`-query also addresses the appropriate situation (same or different passwords).

More precisely, we have proven that for any environment, its advantage in distinguishing the ideal world (with the ideal functionality from Figure 10) and the real world (with the protocol from Figure 7) is bounded by $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q \times \left(2 \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) + 3 \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{robust}}(t) + 2 \cdot \mathsf{Adv}_{\mathcal{F}}^{\texttt{smooth}} + \mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\texttt{c-s-ps-rand}}\right)$, where q is the number of activated players and t its running time.

D.4 Proof of Theorem 8 (UC-Secure OT from an SPHF-Friendly Commitment)

To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$. We prove the adaptive version of the protocol. The proof of the static version can be obtained by removing the parts related to adaptive version from the proof below.

Essentially, one first makes the setup algorithm additionally output the trapdoor (setup-indistinguishability); one can then replace all the commitment queries by simulated (fake) commitments (simulation-indistinguishability). When the sender submits the values $(\mathsf{hp}_i, M_i)_i$ the simulator can extract all the message thanks to the trapdoor and get the witnesses for each indices. This allows to simulate the `Send`-query to the ideal functionality. Eventually, when simulating the honest senders, the simulator extracts the committed value s, to set hp_s and M_s consistent with m_s, the other values can be random. More details follow:

Game $\boldsymbol{G}_0$: This is the real game.

Game $\boldsymbol{G}_1$: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs $(m_1, \ldots, m_k)$ and s sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability*. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: We first deal with **honest senders** P_i: when receiving a commitment C, the simulator extracts the committed value s. Instead of computing the key K_t, for $t = 1, \ldots, k$ with the hash function, it chooses $K_t \xleftarrow{\$} G$ for $t \neq s$.

With an hybrid proof, applying the smoothness (see Figure 6 – left), for every honest sender, on every index $t \neq s$, since C is extracted to s, for any $t \neq s$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased. This game is thus indistinguishable from the previous one under the smoothness.

Game G_4: Still in this case, when receiving a commitment C, the simulator extracts the committed value s. Instead of proceeding as the sender would do on $(m_1, \ldots, m_k)$, the simulator proceeds on $(m'_1, \ldots, m'_k)$, with $m'_s = m_s$, but $m'_t = 0$ for all $t \neq s$. Since the masks K_t, for $t \neq s$, are random, this game is perfectly indistinguishable from the previous one.

Game G_5: We now deal with **honest receivers** P_j: we replace all the commitments $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ in Step 1 of the index query phase of honest receivers by simulated commitments $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, s)$, which means $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, s)$. We then store (ℓ, s, C, δ) in Λ.

With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{s-sim-ind}}$ security game for each session, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value s.

Game G_6: We deal with **the generation of R for honest senders P_i on honestly-generated queries (adaptive case only)**: if P_i and P_j are honest at least until P_i received the second flow, the simulator sets $R = F(S')$ for both P_i and P_j, with S' a random value, instead of $R = F(S)$.

With an hybrid proof, applying the `IND-CPA` property for each session, one can show the indistinguishability of this game with the previous one.

Game G_7: Still in the same case, the simulator sets R as a random value, instead of $R = F(S')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game G_8: We now deal with **the generation of K_s for honest senders P_i on honestly-generated queries**:

- in the static case (the pre-flow is not necessary, and thus we assume $R = 0$) the simulator chooses $K_s \xleftarrow{\$} G$ (for $t \neq s$, the simulator already chooses $K_t \xleftarrow{\$} G$), where s is the index given by the ideal functionality to the honest receiver P_j.

  With an hybrid proof, applying the pseudo-randomness (see Figure 6 – right), for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitments information δ for C;
- in the adaptive case, and thus with the additional random mask R, one can send a random M_s, and K_s can be computed later (when P_j actually receives its flow).

  As above, but only of P_j has not been corrupted before receiving its flow, the simulator chooses $K_s \xleftarrow{\$} G$. With an hybrid proof, applying the pseudo-randomness (see Figure 6 – right), for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitments information δ for C. If the player P_j involved in the pseudo-randomness game gets corrupted (but δ is unknown) we are not in this case, and we can thus abort it.

  In case of corruption of P_i, everything has been erased. In case of corruption of the receiver P_j, and thus receiving the value m_s, the simulator chooses R (because it was a random value unknown to the adversary and all the other K_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta_s) \oplus M_s = m_s.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game G_9: Still in this case, the simulator proceeds on $(m'_1, \ldots, m'_k)$, with $m'_t = 0$ for all i. Since the masks $K_t \oplus R$, for any $t = 1, \ldots, k$, are independent random values (the K_t, for $t \neq s$ are independent random values, and K_s is also independently random in the static case, while R is independently random in the adaptive case), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index s given by the ideal functionality to the honest receiver P_j, to simulate P_i (but it is still necessary to simulate P_j).

Game G_{10}: We do not use anymore the knowledge of s when simulating an **honest receiver** P_j: the simulator generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the index query phase of honest receivers. It then stores $(\ell, \perp, C, \mathsf{eqk})$ in Λ. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game since δ is never revealed. Λ is first filled with $(\ell, \perp, C, \mathsf{eqk})$, it can be updated with correct values in case of corruption of the receiver.

When it thereafter receives $(\mathtt{Send}, \mathrm{sid}, \mathrm{ssid}, P_i, P_j, (\mathsf{hp}_1, M_1, \ldots, \mathsf{hp}_k, M_k))$ from the adversary, the simulator computes, for $i = 1, \ldots, k$, $\delta_i \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, i)$, $K_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\ell, L_i), C, \delta_i)$ and $m_i = K_i \oplus R \oplus M_i$. This provides the database submitted by the sender.

Game G_{11}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionnality, which means that an honest sender has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent an index query, the simulator generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ and R a random value, to send C and c during the index query phase of the honest receiver;
- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value s, and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as S, and computes $R = F(S)$);
- when receiving $(\mathsf{hp}_1, M_1, \ldots, \mathsf{hp}_k, M_k)$ from the adversary (a corrupted sender), the simulator computes, for $i = 1, \ldots, k$, $\delta_i \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, i)$, $K_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, L_i, (\ell, C), \delta_i)$ and $m_i = K_i \oplus R \oplus M_i$. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with m_s, on behalf of a corrupted receiver, from the extracted s, instead of proceeding as the sender would do on $(m_1, \ldots, m_k)$, the simulator proceeds on $(m'_1, \ldots, m'_k)$, with $m'_s = m_s$, but $m'_i = 0$ for all $i \neq s$;
- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(m'_1, \ldots, m'_k)$, with $m'_i = 0$ for all i, but it chooses R uniformly at random instead of choosing it as $R = F(S)$; in case of corruption afterward, the simulator will adapt R such that $R \oplus \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta_s) \oplus M_s = m_s$, where m_s is the message actually received by the receiver.

Any corruption either reveals s earlier, which allows a correct simulation of the receiver, or reveals $(m_1, \ldots, m_k)$ earlier, which allows a correct simulation of the sender. When the sender has sent his flow, he has already erased all his random coins. However, there would have been an issue when the receiver is corrupted after the sender has sent is flow, but before the receiver receives it, since he has kept δ_s: this would enable the adversary to recover m_s from M_s and hp_s. This is the goal of the epheremal mask R that provides a secure channel.

As a consequence, the distance between the first and the last games is bounded by

$$\begin{aligned}
&\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t) + q_s \left(\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathcal{E}}(t) + \mathsf{Adv}^{\mathtt{prg}}_{F}(t) + \mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{C}}(t) + (k-1)\mathsf{Adv}^{\mathtt{c\text{-}smooth}}_{\mathcal{C},\mathcal{F}}(t) + \mathsf{Adv}^{\mathtt{c\text{-}ps\text{-}rand}}_{\mathcal{C},\mathcal{F}}(t) \right) \\
&\leq \mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t) + q_s \times \left(\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathcal{E}}(t) + \mathsf{Adv}^{\mathtt{prg}}_{F}(t) + 2 \cdot \mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{C}}(t) + k \cdot (\mathsf{Adv}^{\mathtt{robust}}_{\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{smooth}}_{\mathcal{F}}) \right),
\end{aligned}$$

where q_s is the number of concurrent sessions and t the running time of the distinguisher.

E Concrete Instantiations

UC-Secure Password-Authenticated Key Exchange. A concrete instantiation of our generic PAKE construction from Figure 7 is presented in Figure 11.

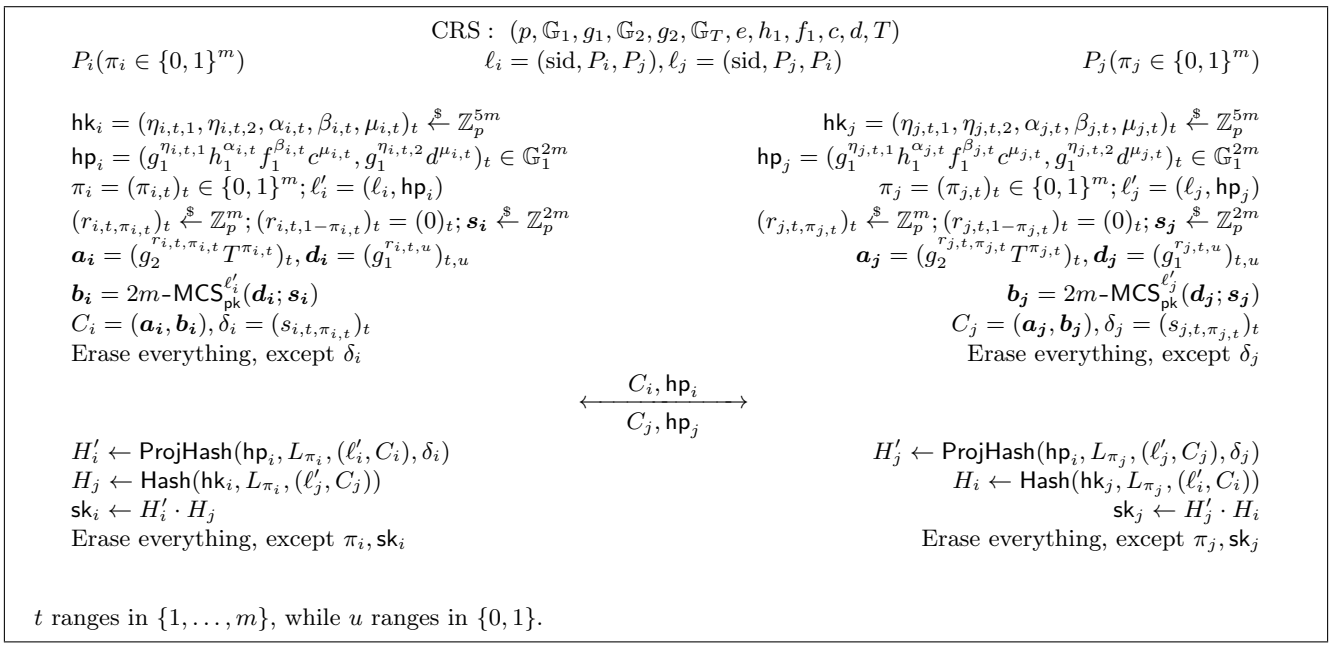

Fig. 11. UC-Secure PAKE

UC-Secure Oblivious Transfert Protocol. A concrete instantiation of our generic OT constructions from Figure 8 is presented in Figure 12. No PRG is actually required here, and we can choose $S \xleftarrow{\$} \mathbb{G}_1$ and $R = e(S, g_2)$. The first flow and the ciphertext c needs not to be sent if only static security is required (in this case $R = 1$).

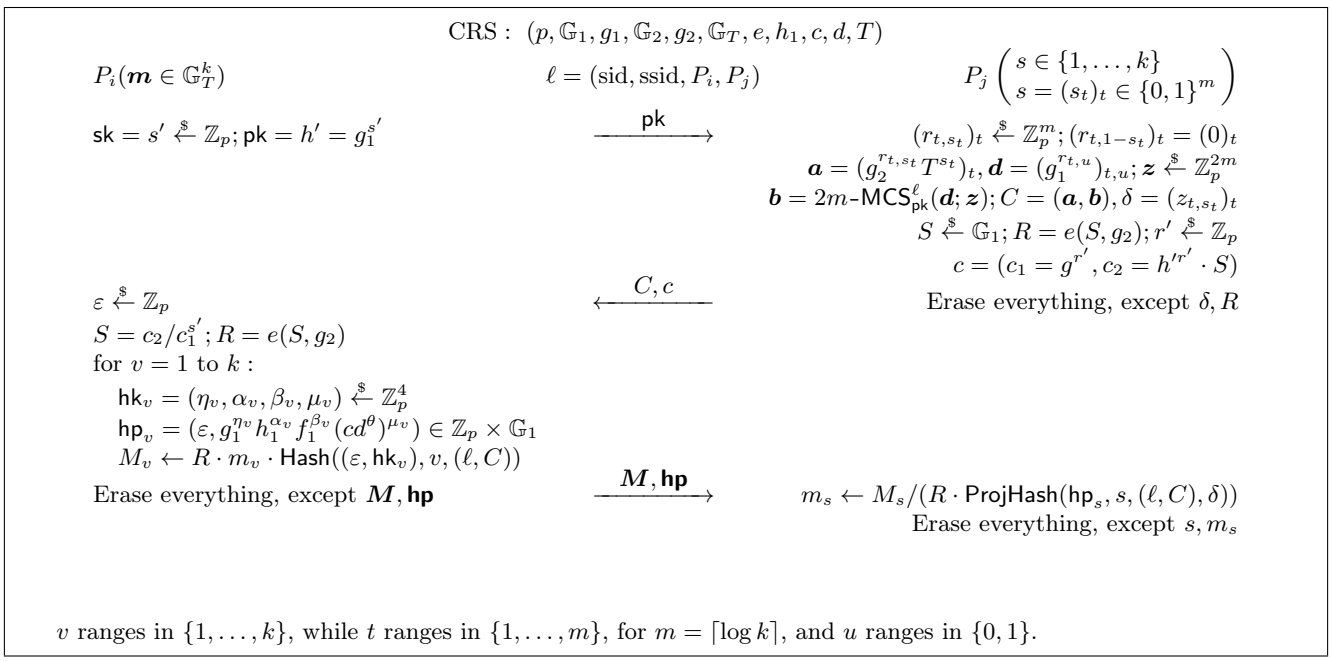

Fig. 12. UC-Secure 1-out-of-k OT Protocol