Efficient and Provably Secure Signature Schemes in the Standard Model

Sven Schäge

Dissertation zur Erlangung des Grades eines Doktor-Ingenieurs der Fakultät für Elektrotechnik und Informationstechnik an der Ruhr-Universität Bochum
Bochum, Mai 2011

Author Contact Information:
sven.schaege@nds.rub.de

Thesis Advisor: Prof. Jörg Schwenk
Thesis Reader: Prof. Eike Kiltz
Oral Examination: 17.06.2011

Abstract

A digital signature scheme is the electronic realization of a classical signature: Alice can produce signatures on documents of her choice, whereas everyone can verify that these documents have indeed been signed by Alice. Besides encryption systems, digital signature schemes constitute the most important primitive of modern cryptography and an indispensable ingredient of electronic business. To be useful in practical applications, it is important that digital signature schemes are secure and efficient at the same time. This thesis presents four results on digital signature schemes which address this challenge: a new and efficient signature scheme that is secure under the Strong Diffie-Hellman assumption; tight security proofs for two large classes of signature schemes that are secure under the Strong Diffie-Hellman assumption or the Strong RSA assumption; new security notions, transformations and constructions of two-tier signature schemes; and a new and efficient ring signature scheme that is solely secure under the Computational Diffie-Hellman assumption. Security of all of these schemes is proven in the standard model while relying on well-analyzed factoring-type and DL-type complexity assumptions.

Zusammenfassung

Digitale Signaturschemata stellen das elektronische Pendant zu klassischen Signaturen dar: Alice kann zu beliebigen Dokumenten eine Signatur erstellen. Mit Hilfe dieser Signatur kann öffentlich überprüft werden, dass das entsprechende Dokument tatschlich von Alice unterzeichnet wurde. Die Sicherheitseigenschaften eines Signaturschemas müssen dabei insbesondere garantieren, dass niemand außer Alice gültige Signaturen erzeugen kann, welche auf Alice als Urheber hinweisen. Neben Verschlüsselungssystemen gehören digitale Signaturverfahren zu den wichtigsten Bausteinen der modernen Kryptografie und stellen ein unverzichtbares Werkzeug für den elektronischen Geschäftsverkehrs dar. Im praktischen Einsatz ist wichtig, dass sie nicht nur starken Angriffen widerstehen können sondern auch gleichzeitig hohe Effizienz bieten. Die vorliegende Arbeit präsentiert vier Ergebnisse aus dem Forschungsgebiet der effizienten und beweisbar sicheren digitalen Signaturen: ein effizientes digitales Signaturschema, das sicher unter der Strong Diffie-Hellman Annahme ist; neue und effizientere Sicherheitsreduktionen für zwei allgemeine Klassen von digitalen Signaturschemata; neue Sicherheitsdefinitionen, Transformationen und Konstruktionen von Two-Tier Signaturschemata; und ein neues und besonders effizientes Ringsignaturschema, welches nur auf der Computational Diffie-Hellman Annahme basiert. Alle betrachteten Schemata sind sicher im Standardmodell und basieren auf schwachen und gut untersuchten Komplexitätsannahmen.

Acknowledgments

I would like to thank my supervisor Jörg Schwenk for supporting me, believing in me, and giving me the freedom to pursue my research interests. My thanks also go to my thesis reader Eike Kiltz for giving me invaluable advice and support. I would like to thank all previous and present members of the Horst Görtz Institute for IT-Security for contributing to a relaxed atmosphere. Special thanks go to all members of the Chair of Network and Data Security with whom I had a lot of productive, interesting, funny, and entertaining discussions. I feel very privileged for having them as my colleagues. In particular I would like to thank my office-mate Tibor Jager for very interesting and fruitful discussions, his moral support, and for being a very good friend and companion on our journey into the theoretical foundations of cryptography. I would also like to thank Mathias Herrmann and Florian Kohlar who have become very good friends to me. Finally I would like to thank my beloved wife Behnas and our wonderful daughter Elina Niloufar. There are no words to express my gratitude for their support and love.

Contents

Chapter 1

Introduction

In a digital signature scheme Alice, the signer, can use her secret key to sign arbitrary messages. The output of such a process is called signature. Given a signature, Alice's public key and the message anyone can verify that the signature is valid, i.e. it has actually been produced on this message with the help of her secret key. The security properties of the scheme have to guarantee that nobody is able to produce valid signatures without Alice's secret key. In this way, a signature proves that Alice is the true originator of the signature.

Besides public key encryption systems, digital signature schemes are the most important primitive of modern cryptography. They provide user and message authentication, message integrity, and constitute the only cryptographic constructions that allow to convince anyone (and not only a designated receiver) that a signed message has been sent by a certain user. This gives a digital mechanism that irreversibly binds the signer to the signed document (non-repudiation) – an important pre-requisite for contract signing and digital business. Their security properties make digital signature schemes a key building block in a plethora of higher level protocols and the most important tool for constructing trust-hierarchies like the public key infrastructure (PKI) or the PGP web of trust.

Digital signature schemes belong to the realm of public-key cryptography that since the seminal works of Diffie and Hellman [46] (1976) and Rivest, Shamir and Adleman [90] (1978) first came into the focus of (public) scientific attention. Since then research on this field has made tremendous progress.

1.1 Scope

In this thesis, I 1) present new and 2) improve existing signature schemes which provide high security guarantees under weak security assumptions while being highly efficient. Such schemes make signature-based applications more attractive as fewer additional resources are required to implement security mechanisms. At the same time they foster confidence in these applications as security only relies on very weak and well-analyzed security assumptions. My results encompass many practical and theoretical aspects of digital signature schemes ranging from very theoretical topics like new security definitions, transformations, and security proofs to rather practical issues like efficiency improvements of existing signature schemes.

In this thesis, my main focus lies on digital signature schemes. However, the results I obtained in my PhD phase at Ruhr-Universität Bochum broadly deals with secure authentication mechanisms (which signature schemes are an important part of). I also worked on authentication methods and applications of signature schemes in higher-level applications like cloud computing, e-voting, and key-agreement protocols.

1.2 Publications

Below, I provide a list of my relevant results.

Refereed Conference Proceedings

Tight Proofs for Signature Schemes without Random Oracles **2011**

- Sven Schäge. Accepted for EUROCRYPT 2011, Tallinn, May 15th-19th, 2011. Springer, LNCS, 2011.

Generic Compilers for Authenticated Key Exchange **2010**

- Tibor Jager, Florian Kohler, Sven Schäge, Jörg Schwenk. ASIACRYPT 2010, Singapore, December 5th-9th, 2010. Springer, LNCS 6477, 2010.

Towards an Anonymous Access Control and Accountability Scheme for Cloud Computing **2010**

- Meiko Jensen, Sven Schäge, Jörg Schwenk. 3rd International Conference on Cloud Computing (CLOUD), Miami, Florida, USA, July 5th-10th 2010. IEEE Computer Society, 2010.

A New RSA-Based Signature Scheme **2010**

- Sven Schäge, Jörg Schwenk. AFRICACRYPT 2010, Stellenbosch, South Africa, May 3th-6th, 2010. Springer, LNCS 6055, 2010.

A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys **2010**

- Sven Schäge, Jörg Schwenk. Financial Cryptography Fourteenth International Conference, FC 2010, Tenerife, Spain, January 25th-28th, 2010. Springer, LNCS 6052, 2010.

Twin Signatures, Revisited **2009**

- Sven Schäge. Provable Security Third International Conference, ProvSec 2009, Guangzhou, China, November 11th-13th, 2009. Springer, LNCS 5848, 2009.

1.3 Previous Publication

This thesis presents four of my main results.

- The signature scheme of Chapter 3 originally appeared in the proceedings of Provable Security 2009 as “Twin Signature Schemes, Revisited”.
- The results on tight security proofs given in Chapter 4 originally appeared in the proceedings of EUROCRYPT 2011 as “Tight Proofs for Signature Schemes without Random Oracles” and were presented in May 2011.
- The results on two-tier signature schemes of Chapter 5 have been obtained in very recent work and will be submitted to an international conference soon.
- The ring signature scheme of Chapter 6 originally appeared in “A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys”, joint work with Jörg Schwenk, which was presented at Financial Cryptography 2010.

In all these works, I am the main (or only) contributor.

1.4 Overview

The first chapter of this thesis is devoted to a broader introduction to digital signature schemes. It gives an overview over general issues like feasibility results, security models, and construction principles of digital signature schemes. Of particular importance is Section 1.6 as it sums up important research directions and challenges in research on digital signature schemes. The second chapter formally introduces the cryptographic primitives, definitions, and complexity assumptions required in this thesis. Chapter 3 to Chapter 6 present four of my main results on digital signature schemes. Each chapter is preceded by a brief introduction and where adequate with additional information on related work. Chapter 7 sums up and interprets the results of this thesis before it turns to a brief outlook on future research directions.

1.5 Provably Secure Signature Schemes

In the following, we provide an overview on the existing results on digital signature schemes and provide background information on security models, security assumptions, security proofs, and security transformations.

1.5.1 General Results

In a series of works initiated by Lamport [71], Bellare-Micali [5], Naor- Yung [82], and Rompel [92] showed how to construct signature schemes from one-way functions. From a complexity theoretic point of view, this is the best result available since signature schemes trivially give rise to one-way functions (in fact the key generation algorithm of a signature scheme – when restricting its output to only consist of the public key – always constitutes a one-way function). We note that all of the above results rely on tree-based structures of signature schemes (see below). Before we can deal with construction principles of signature schemes in more detail, we have to introduce the security model that underlies our analyses.

1.5.2 Security Model: Standard Model

In this thesis we concentrate on schemes which are secure in the so-called standard model (or plain model). In this model, the adversary's resources are bounded polynomially and security is based solely on the underlying complexity assumption. As a result, the standard model requires very weak set-up assumptions. However, there are also models in cryptography which assume that all parties have access to a certain cryptographic primitive in its idealized form. The two most important models in this category are the random oracle model and the generic group model. Both have frequently been used for the design of provable digital signature schemes.

The Random Oracle Model. The random oracle methodology is a general approach to argue about the security properties of cryptographic schemes [7]. First a cryptographic scheme is proven secure in an ideal model, the random oracle model (ROM), where all parties have access to a function – the *random oracle* – that outputs truly random bit strings. In practical realizations of the scheme, the random oracle is substituted by a cryptographic hash function. The idea behind this substitution is that good cryptographic hash functions should behave like random functions (oracles).

The Fiat-Shamir Heuristic. In 1986, Fiat and Shamir proposed a general way to produce signature schemes from secure (3-move) identification protocols [53]. Roughly, the signer acts almost exactly like the prover in the identification protocol. The only difference is that it now computes the challenge values, which are originally drawn at random by the verifier, using a random oracle and the message to be signed. The Fiat-Shamir heuristic is a very general methodology to obtain efficient signature schemes (also called signatures of knowledge [37]) that can be based on standard security assumptions like the factoring assumption or the discrete logarithm assumption.

Critics. The random oracle model represents a very strong idealization of the real world and in 1998, Canetti, Goldreich and Halevi [32] showed that there exist (rather artificial) signature schemes which have a security proof in the random oracle model but are insecure when instantiated with any hash function. Subsequent results showed that such examples also exist for more practical signature schemes [33, 60].

Since these seminal results cryptographic research focused more and more on efficient cryptographic schemes that are solely secure in the standard model which does not assume any primitive to be available in its idealized form. However, most of the existing signature schemes are still only secure in the random oracle model while there exists only few signature schemes that are secure in the standard model.

Generic Models. Another model with special setup assumptions is the generic group [97, 76] model that was later extended to also cover rings (generic ring model) [72]. In these models all parties are only provided with idealized versions of the underlying mathematical structures (like groups or rings) in which the corresponding cryptographic schemes are defined in. The generic models have been used to study the security of several complexity assumptions like RSA [1] or the q-Strong Diffie-Hellman [16] assumption. However, similar to the random oracle model, positive results obtained in the generic models cannot be regarded as sound cryptographic security proofs. In fact, there exist popular examples of computational tasks which are provably as hard as some underlying well-analyzed number-theoretical problem in the generic model but which is very easy to solve when using concrete instantiations of rings or groups [68].

1.5.3 General Types of Signature Schemes

The existing signature schemes fall in one of two categories according to their underlying construction principle: tree-based signature schemes or hash-and-sign signature schemes. Tree-based signature schemes can be based on very general assumptions while hash-and-sign schemes generally provide better efficiency.

Tree-based Signature Schemes. Besides the general results on signature schemes of [71, 5, 82, 92], several works have presented tree-based signature schemes that rely on more concrete security assumptions like the factoring, RSA, and Computational Diffie-Hellman assumption. Among these works are important results by Goldwasser-Micali-Rivest [61], Goldreich [59], Merkle [77], Dwork-Naor [48], Cramer-Damgard [42, 43], Boneh-Mironov-Shoup [23], and Catalano-Gennaro [34]. The basic idea of tree-based signatures is to stepwisely construct a (binary) tree where each node can be used to authenticate (sign) a single message. Additionally, each parent node authenticates its children. A signature basically consists of a path, i.e. a set of chain-authenticated nodes, from the message to the root node which in turn serves as the scheme's public key. The signature size is thus dependent on the depth of the tree. Several of the above results show how to construct shorter tree-based signatures by allowing for larger branching factors of the tree at the cost of a slight increase in the public keys size. In essence, they exploit a trade-off where the tree depth can be decreased logarithmically if the size of the public parameters is increased linearly. Recently, the tree-based construction principle was used to build leakage-resilient signature schemes from leakage-resilient chameleon hash functions [52]. For more details we refer the original papers. In particular [23] gives a good overview on tree-based signatures.

Hash-and-Sign Signature Schemes. Hash-and-sign signatures can be shorter than tree-based signatures as they do not require signature schemes to consist of a linear number (in the bit size of the input message) of elements. Today, there exist many realizations of hash-and-sign signature schemes in the random oracle model like [50, 94, 84, 8, 88, 22, 58, 57].

In 1999, Gennaro, Halevi and Rabin [56] and independently Cramer and Shoup [45] presented the first hash-and-sign signature schemes that are secure in the standard model. Both schemes are secure under the Strong (or Flexible) RSA assumption [3]. Subsequent works presented further schemes that rely on the Strong RSA assumption. Except for the work by Naccache, Pointcheval, and Stern [80], these schemes either provide better efficiency than [45] like the Zhu [103, 104] and Fischlin [54] signature scheme or are very suited for the design of protocols that prove knowledge of a valid signature without revealing it like the scheme by Camenisch-Lysyanskaya [30].

Starting with Boneh and Franklin in 2001 [19], the (constructive) use of bilinear groups in cryptography paved the way for signature schemes with very short signature size. The decisive advantage of bilinear groups is the very compact representation of group elements. In 2004, Boneh and Boyen presented the first hash-and-sign signature scheme that makes use of bilinear groups [17]. The Boneh-Boyen signature scheme was proven secure under a new flexible assumption, the q-Strong Diffie Hellman (SDH) assumption. In 2004, Camenisch and Lysyanskaya also presented a signature scheme that relies on bilinear groups [31]. Unlike the Boneh-Boyen scheme, their scheme is proven secure under the LRSW [75] assumption. However, in the same paper Camenisch and Lysyanskaya propose a variant that is based on the SDH assumption in bilinear groups. The corresponding security proof was provided two years later in [85, 2]. In 2005, Brent Waters presented a signature scheme (as a by-product of his identity-based encryption system) that is solely secure under the Computational Diffie-Hellman assumption in bilinear groups [102]. In 2008, Hofheinz and Kiltz presented two new signature schemes, an SRSA-based signature scheme and an SDH-based scheme [64]. The SDH based signature scheme delivers the shortest standard model signatures so far. In 2009, Waters and Hohenberger presented the first signature scheme in the standard model which is solely secure under the RSA assumption and so solved a long-standing open problem [66].

Stateless versus Stateful Signature Schemes. Several of the early signature schemes like those by Merkle [77] and Naor-Yung [82] and even some very recent schemes like the one by Hohenberger-Water [65] are stateful, i.e. the signer has to store some data that is refreshed after each signing procedure. In tree-based signature schemes this state is often used to help the signer keep track of which nodes of the tree have already been used for signing. This is crucial because if the signer would use a single leaf twice for signature generation, the adversary could use the corresponding signatures to generate new signatures on fresh messages. For this reason it is very important to protect the signers state from adversarial manipulations. As a consequence, implementing a stateful signature scheme requires much more effort than stateless schemes: not only the secret key (which is fixed after the initial generation and thus can be implemented using secure read-only memory) must be protected against tampering attacks but also a continually updated state (which requires secure read/write memory).

In 1986, Goldreich showed how to turn stateful tree-based signature schemes into stateless schemes using pseudo-random functions [59]. Applying his results to the Goldwasser-Micali-Rivest scheme [61], Goldreich so obtained the first stateless signature scheme which is provably secure in the standard model. However, this transformation requires a blowup in the number of leafs of the tree from t to t^2.

1.6 Comparison of Signature Schemes

Whether or not a signature scheme is useful for a certain application depends on several factors. In this section we give a brief overview on the most important features.

1.6.1 Efficiency

One of the most important characteristics of a signature scheme is the size of the transmitted data (per message).

As a public key must only be transmitted once for each signer, this size is typically dominated by the length of a signature. However, in scenarios where a signer often has to retrieve new public keys the size of the public key plays an important role too. This is especially important when, besides the message and the signature, the verifier must also process a chain of certificates.

The size of the scheme's parameters in turn is related to the underlying complexity assumption and the quality of the security reduction. The parameter lengths are chosen such that if an adversary would be able to break the security of the signature scheme it would also be able to break a very hard instance of the complexity problem that underlies our security assumption.

The signature size is of course also related to the signing and verification complexity. The larger the size of the concrete parameters of the signature scheme.

We note that in some applications the response time of the involved parties (signing and verification complexity) may be very import. In such systems it may be useful to employ additional mechanisms to speed up the time complexities as batch verification [4] or offline/online signatures [51] see 1.6.4.

1.6.2 Tightness Of Security Proofs

In cryptography, a security proof typically has the form of a security reduction. To show that a digital signature scheme S is secure under complexity assumption C, one has to show that any successful PPT attacker $\mathcal{A}$ against S can be used (in a black-box-way) to build a successful attacker $\mathcal{B}$ – the simulator – that breaks the complexity assumption. To prove asymptotic security it is only necessary to show that $\mathcal{B}$ is a probabilistic polynomial time algorithm. However, following the exact (or concrete) security approach introduced by Bellare, it is very useful to also quantify these reductions [9]. This helps to gain estimates on the required parameter sizes of the scheme when aiming for a certain security level. Let us go into more detail.

A typical formulation of a security assumption says that a given problem at hand cannot be solved by any polynomially bounded adversary that runs in time t with success probability greater than ϵ. In a similar way, the formal definition of a secure signature scheme states that no t'-time adversary is able to produce a forgery with probability ϵ' while making q queries to the signing oracle. Of course, in both cases the running time and the success probability depends on the size of the problem that an adversary must solve (i.e. the size of the input parameters available as described in the description of C, or the system parameters of S). The larger the size, the harder the problem. To account for this, all parameters (respectively their sizes) are described as functions of the security parameter κ.

The tightness of the security reduction is a quality criterion for constructions of signature schemes. Surely we have that the running time of $\mathcal{B}$ is greater than that of $\mathcal{A}$ because $\mathcal{B}$ is used by $\mathcal{A}$ in a blackbox manner. Thus, $t' \geq t$. On the other hand in case $t' = t$, $\mathcal{B}$'s success probability ϵ' cannot be greater than that of the success probability of $\mathcal{A}$: $\epsilon' \leq \epsilon$. If $\epsilon \approx \epsilon'$ while $t \approx t'$ the security reduction is called tight. Otherwise, for example if $\epsilon' \leq \epsilon^2$ or $\epsilon' \leq q\epsilon$ the reduction is said to be loose. It is important to note that, at the same level of security, loose security reductions require the scheme's parameters to be larger than tight reductions. Section 4.4 provides a detailed example of this effect.

1.6.3 Weaker Security Assumptions

Security of most signature schemes relies on complexity (security) assumptions. Usually these assumptions belong to a broader category of assumptions like factoring-based, discrete logarithm based, or lattice-based assumptions. The schemes presented in this thesis all rely on factoring-based or discrete logarithm based complexity assumptions.

Cryptography has always been interested in finding relations between these assumptions. Some seemingly different assumptions have been proven to be equal but in most cases the best one can hope for is to show that one assumption implies another. As an example we mention the factoring, the RSA, and the SRSA assumption. It is well known that any attacker that breaks the factoring assumptions is able to break the RSA assumption. However, the converse is not known to be true. Correspondingly the SRSA assumption is known to imply the RSA assumption, as any successful attacker against the RSA assumption can be used to break the SRSA assumption. Again the converse is not known to be true (at least not using a reduction with a security loss of at least a polynomially, in the security parameter, large factor). In complexity theoretic terms, such implications show that the factoring assumption is *weaker* than the RSA (or the RSA assumption is *stronger* than the factoring assumption). Similarly the Strong RSA assumption, as the name suggests, is stronger than the RSA assumption.

Intuitively, weaker security assumptions provide higher security guarantees. This is because a system that is secure under a very weak security assumption (like factoring) can remain secure even if stronger security assumptions have been broken. It is one of the most important tasks of cryptography to devise new and efficient schemes that rely on very weak security assumptions (or at least weaker than in state-of-the-art solutions).

Unfortunately, it seems very hard to prove that the above assumptions actually hold. All we have today is evidence. This is founded on two research directions. The first one is cryptanalysis, where researches devise new attacks that have better running time, space efficiency or success probability than previous solutions. Good security assumptions have withstood all attacks that have been proposed so far. Second, a considerable amount of work has been invested in analyzing security assumptions in restricted models (e.g. generic models) of computations (see Section 1.5.2). Such models assume that the attacker is only provided with the generic operations of the underlying mathematical structure. For example, he may not exploit knowledge of the bit representation of a given group or ring element. However, as Jager and Schwenk showed, results in generic models may not have implications for real world implementations [68]. They showed that the computation of the Jacobi symbol, which is efficiently feasible in practice, is equivalent to factoring in the generic ring model. So problems which are provably hard in a generic model may be trivially solvable in practice.

Besides the analysis of relations between security assumptions theory says little about the quality of a concrete security assumptions. Cryptographers prefer security assumptions which are very simple and well-established (i.e. well-analyzed by security experts over a longer period of time). Also, security assumptions with a static amount of input elements seem to be more attractive than schemes where the size of the set of input values given to the attacker is polynomially dependent on the security parameter (so-called q-type assumptions, like SDH). Finally, cryptographers prefer to base their schemes on hardness problems where the attacker must output a single solution for each instance (like for example the factoring problem or the RSA problem). This stands in contrast to problems where the attacker may output one of an exponentially large set of solutions (like SRSA).

1.6.4 Response Time: Offline/Online Signatures and Batch Verification

In real-time applications, the response time of the involved parties may be very import. On the verifier's side this can be done by employing signatures that allow for efficient batch verification [4]. In such schemes, signatures are first combined with each other. The resulting combination is then input to a modified verification procedure. In this way several signature schemes can be verified at the same time. Batch verification pays off whenever the initial combining of signature schemes can be accomplished very efficiently when compared to the original verification procedure. This makes batch verification particularly useful in pairing based signature schemes, as the application of the bilinear pairings typically dominates the costs of the verification process. Instead of applying the pairing separately for each signature, it is only applied to a combination of signatures. This process relies on homomorphic properties of the produced signatures and cannot applied to signature schemes in general. Batch verification can also be applied to more complex variants of signature schemes like ring signature schemes and group signature schemes [29].

On the signer's side, response time (signing complexity) can be improved by employing offline/online signature schemes [51]. In such schemes the bulk of the work for producing a signature can be done in advance, for example in times when the signer's computational device is not working to full capacity. Obviously, this work must be message-independent. When the signer finally wants to sign a message the remaining work load is very small. Many existing signature schemes give rise to offline/online variants in a straight-forward manner. In particular signature schemes that are constructed using signature transformations inherently support such an operation mode. Typically the remaining workload reduces to finding a collision for a chameleon hash function. However, usually offline/online signatures require a small growth in the size of the secret key as compared to the original scheme, for example the secret key of the chameleon hash function is additionally needed in the signing process.

1.7 Constructions of Signature Schemes from Complex Primitives

The signature schemes obtained using a general construction from one-way function are too inefficient to be used in practice. However, one can construct efficient and secure signature schemes from *more complex* primitives as well. As documented by Boneh and Franklin in [19], Moni Naor observed that secure signature schemes can directly be obtained from identity-based encryption systems. Popular examples of signature schemes which follow this line of construction are the very short signature scheme by Boneh, Lynn, and Shacham (which is secure in the ROM) [21] and the CDH-based signature scheme by Brent Waters that is secure in the standard model [102].

1.8 Signature Transformations - Transformations from Signature Schemes with Weak Security Properties to Schemes with Stronger Properties

Building efficient and secure signature schemes in the standard model while relying on reasonable security assumptions is a highly non-trivial task. This is because the standard security definition of signature schemes grants the adversary substantial attack capabilities.

What has proven very useful in the past is to define weakened security definitions of signature schemes while, at the same time, delivering a signature transformation that generically maps signature schemes secure under such a weak security definition to signature schemes that are secure in the standard sense. This considerably reduces the complexity in the design process because now the developer can confine himself to building schemes that only fulfill a subset of the security properties required by (fully) secure schemes. Formal security definitions of signature schemes are provided in Section 2.11. Applying the right transformation then (rather automatically) delivers a secure signature scheme without additional design efforts.

It is important to note that it is not reasonable to weaken the standard notion in an arbitrary way. To be actually useful, every new definition must be accompanied by a corresponding signature transformation that allows to securely (and efficiently) construct fully secure schemes from schemes that conform to this new notion. This restricts the set of possible modifications of the security definition. By now, there exists several useful signature transformations. The transformations by Cramer *et al.* [44], Even *et al.* [51], and Shamir and Tauman [96] map signature schemes that are secure under random message attacks to fully secure schemes (we refer to Section 2.11 for the formal security definitions). The results in [51, 96] also show how to build fully secure signature schemes from weakly secure schemes. Huang *et al.* [67] and Bellare and Shoup [13] showed how to construct strongly secure signature schemes from any signature scheme that is secure in the standard sense. Recently, Hohenberger and Waters [66] and Brakerski and Tauman [25] show how to map signature schemes which are universally/selectively secure under generic chosen message attacks to weakly secure schemes.

The importance of this design methodology cannot be overestimated. In fact, many schemes published in the last years have used this approach. And also in the retrospect several existing schemes can be found to implicitly follow this design methodology. Among these schemes are for example the very short signatures by Boneh and Boyen [17] and Hohenberger and Waters [66].

A signature transformation roughly consist of two parts. First it contains a description of how to construct a signature scheme B from a given (weakly secure) signature scheme A. Second it provides a security reduction showing that any attacker $\mathcal{B}$ (as modeled in the standard notion of security) which breaks the security of scheme B can efficiently be used to construct an attacker $\mathcal{A}$ that breaks the security properties of scheme A. The intended conclusion is that, under the assumption that there exist no successful attacker $\mathcal{A}$ against the security of the weak signature scheme A, there cannot exist a successful attacker $\mathcal{B}$ against the constructed scheme B (as it would imply the existence of $\mathcal{A}$).

There are two main criteria to measure the quality of a signature transformation. Probably the most obvious is to analyze the efficiency of the constructed signature scheme B. Of course, it is much more preferable to use transformations where B is only slightly less efficient than the starting scheme A. The second criterion deals with the security reduction of the signature schemes. As sketched before, security reductions should be formulated concretely (i.e. in a quantitative way) rather than asymptotically to find estimates for the necessary parameter lengths in concrete instantiations. As with security proofs based one complexity assumptions, loose reductions require that B uses larger parameter sizes to provide the same level of security as A. Therefore the quality of the security reduction also has immediate consequences for the efficiency of B. Finally, we emphasize that improvements of signature transformations - typically more efficient constructions or tightened security reductions - have a much broader application than improvements to single signature schemes. This is because the obtained improvements transfer to all signature schemes which rely on the corresponding transformation. Thus, developing new or improved signature transformations is of utmost value in cryptography.

1.9 Application of Signature Schemes in Higher-Level Protocols

Digital signature schemes are a valuable tool in many higher-level protocols. Usually, they are used as the primitive of choice for authentication and integrity protection. Examples range from today's most important key-agreement protocol TLS (with server certificate) to cloud computing and identity management systems.

However, there are also variants of signature schemes like two-tier signature schemes (formally introduced in Section 2.12) that provide relatively weak security guarantees while offering highly efficient signing and verification algorithms. Such schemes cannot be used as substitutes for classical signature schemes in most applications. But often they can be used as technical tools to improve the efficiency of more complex protocols. For example, two-tier signature schemes can be used to make several signature transformations more efficient (as shown in Chapter 5).

1.10 Signature Schemes with Special Properties: Privacy-Preserving Signatures

Since the development of the first signature schemes, the basic concept of signature schemes has continually been extended and given rise to signature primitives that fulfill additional security properties. A great portion of these efforts focus on signature schemes with anonymity guarantees for the signer.

Two of the most popular variants of these so-called privacy-preserving signature schemes are ring signature schemes [91, 14] and group signature schemes [6, 12]. The concept of privacy-preserving signatures may sound paradoxical at first, as signature schemes are typically used to uniquely identify the signer of a certain message whiland even prove this relation to third parties. However, a ring signature (or group signature) basically states that the actual signer of a message originates from a well-specified set of possible signers. Verification only allows the verifier recognize that the signer is indeed part of this set while the security properties of the scheme guarantee that the signer cannot be identified among the other members of the set. Although being useful in many scenarios on their own, such schemes are often used as key ingredients in higher-level systems. Applications of these types of signatures range from trusted computing [26] over privacy-preserving identity management systems [28] to the construction of designated verifier signatures [73].

Chapter 2

Preliminaries

2.1 Notation

In cryptography, the provided level of security of a system is usually dependent on the length of the used parameters. To allow for asymptotic analyses, security statements depend on a security parameter $\kappa \in \mathbb{N}$. We use 1^κ to describe the string that consist of κ ones.

Sets, Strings, Intervals. If S is a set we write $|S|$ to denote its size. For a set S, we use $x \overset{\$}{\leftarrow} S$ to denote that x is drawn from S uniformly at random. If s is a string, we write $|s|_2$ to denote its bit-length. For $a, b \in \mathbb{Z},\ a \leq b$ we write $[a; b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$. When dealing with bit strings, we use 0^i (1^i) for $i \in \mathbb{N}$ to denote the string that consists of i zeroes (ones). These strings can be concatenated. For example, we use $1^i 0^j$ to denote the bit string that consists of i ones followed by j zeroes. For clarity, if s is a bit string we may also use $s||s$ to denote the result when concatenating s with itself. For $m_i \in \{0,1\}^{l_m}$ we use $m_i = m_{i,1} \ldots m_{i,l_m}$ to denote the binary representation of m_i with m_{i,l_m} being the most significant bit.

2.2 Prime Numbers

2.2.1 Primes and RSA Moduli

In the following, we use $\mathbb{P}$ to denote the set of primes. We use $\mathbb{P}_{i,j} = \mathbb{P} \cap [2^{i-1}; 2^j - 1]$ with $i, j \in \mathbb{N}\ i \leq j$ to denote the set of primes in $[2^{i-1}; 2^j - 1]$. Observe that for $2 \leq i \leq j$ we have $\mathbb{P} \cap [2^{i-1}; 2^j - 1] = \mathbb{P} \cap [2^{i-1}; 2^j]$.

Prime Number Theorem.

Theorem 2.1 ([93]) *Let $\pi(x)$ denote the number of primes less than or equal to x. For $x \geq 55$, we have that*

$$\frac{x}{\log(x) + 2} < \pi(x) < \frac{x}{\log(x) - 4} \ . \tag{2.1}$$

As a direct application of this theorem we can easily prove the following corollary.

Corollary 2.2 *If $l \geq 7$, $|\mathbb{P}_{l,l}|$ is lower-bound by*

$$|\mathcal{P}_{l,l}| > (2^l - 1)/(\ln(2^l - 1) + 2) - (2^{l-1} - 1)/(\ln(2^{l-1} - 1) - 4). \tag{2.2}$$

2.2.2 Injective Prime Mapping

In one of our signature schemes we will make use of an injective function $\mathsf{toPrime} : \{0,1\}^l \to \mathbb{P}$ for mapping bit strings to primes. We will now present two instantiations of this function.

The first one has been proposed in [80]. It requires a prime mapping function $\mathsf{nextPrime} : \{0,1\}^l \to \mathbb{P}_{1,l+1}$ that maps l-bit strings to the set of primes $\mathbb{P}_{1,l+1}$. Let $l > 2$. Then, the function $\mathsf{nextPrime}$ maps to the smallest prime that is equal or greater than the input value (when interpreted as a natural number). Technically, $\mathsf{nextPrime}$ computes the next odd integer that is greater than or equal to its input value and executes a primality test. If the result of this test is negative $\mathsf{nextPrime}$ simply increments the input value by two. This process is repeated until the result is positive. Observe that, as a direct application of the Bertrand-Chebychev theorem we get that there exists at least one prime greater than 2^l and smaller than 2^{l+1}. Thus, even for the largest input value, $2^l - 1$, there exists an output value in $[2^l; 2^{l+1} - 1]$. (By the prime number theorem this prime is approximately around $l + 2^l$.) Naccache, Pointcheval and Stern define:

$$\mathsf{toPrime}(m) = \mathsf{nextPrime}(m \cdot 2^Q), \tag{2.3}$$

where Q is chosen such that the existence of a prime in any set $[m \cdot 2^Q; (m+1) \cdot 2^Q[$, for $m < 2^l$ is guaranteed. Following [80] this can be realized under reasonable number-theoretic assumptions if $Q \approx 5 \log_2 l$ resulting in about $20l$ primality test (as part of $\mathsf{nextPrime}$).

A second instantiation has recently been proposed by Hohenberger-Waters [66]. In contrast to the Naccache-Pointcheval-Stern method, primes can be much smaller. However, in contrast to the above solution there may not exist an efficient algorithm to compute, given a prime, its pre-image. At the heart of the Hohenberger-Waters solution is a pseudo-random function $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ (for completeness we provide the formal definition of pseudo-random functions in Section 2.7). The key to this function $k_{\mathsf{PRF}} \leftarrow \mathsf{PRF.KeyGen}(1^\kappa)$ is public. Assume $\mathsf{PRF.Eval}$ maps $((\log_2 l)^2 + l)$-bit strings to l-bit strings. Hohenberger-Waters define:

$$\mathsf{toPrime}(m) = \mathsf{PRF.Eval}(k_{\mathsf{PRF}}, i||m), \tag{2.4}$$

where $i \in [0; (\log_2 l)^2]$ is called the resolving index of m which is the minimal value in $[0; (2\log_2 l)^2]$ such that the output of $\mathsf{PRF.Eval}(k_{\mathsf{PRF}}, i||m)$ is prime. Hohenberger showed that the probability of any PPT to find two colliding input values is negligible given that the pseudo-random function is secure as defined below. For a proof we refer to the original paper [66]. We note that there exists efficient pseudo-random functions which solely rely on standard assumptions like the factoring assumptions.

2.2.3 Safe RSA Moduli

Definition 2.3 (Safe Prime) *Let p be an odd prime with $p = 2q + 1$ for some $q \in \mathbb{N}$. If q is prime, p is called a safe prime.*

Definition 2.4 (Safe Composite) *Let n be a composite and $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ be its prime factor decomposition. If all the p_i for $i \in [1; l]$ are safe primes, n is called a safe composite (or safe modulus).*

Definition 2.5 (Balanced Primes) *Let $p, q \in \mathbb{N}$. We say that p and q are balanced if $|p|_2 = |q|_2$.*

Definition 2.6 (RSA Modulus) *Let $p, q \in \mathbb{N}$ be distinct and balanced primes. Then we call $n = pq$ an RSA modulus. If n is a safe composite, we say that n is a safe RSA modulus.*

Definition 2.7 (Quadratic Residue) *Let n be an RSA modulus. We call $\mathcal{QR}_n = \{x | \exists y \in \mathbb{Z}_n^* : y^2 = x \bmod n\}$ the set of quadratic residues (modulo n).*

2.2.4 Bilinear Groups and Secure Bilinear Maps

Definition 2.8 (Bilinear groups) *Let $\mathbb{G}_1 = <g_1>$, $\mathbb{G}_2 = <g_2>$ and $\mathbb{G}_T$ be groups of prime order p. The function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is called bilinear pairing (or bilinear map) if it holds that*

1. *$\forall a \in \mathbb{G}_1, b \in \mathbb{G}_2, x, y \in \mathbb{Z}_p : e(a^x, b^y) = e(a, b)^{xy}$ (bilinearity),*
2. *$e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ is a generator of $\mathbb{G}_T$ (non-degeneracy), and*
3. *e is efficiently computable (efficiency).*

We call $B = (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ (asymmetric) bilinear group. If $\mathbb{G}_1 = \mathbb{G}_2$ we say that $B = (\mathbb{G}_1, g_1, \mathbb{G}_T, p, e)$ is a symmetric bilinear group.

2.3 Complexity Assumptions

The signature schemes presented in the thesis rely on security assumption which fall into one of two classes: factoring-based or discrete logarithm based assumptions.

2.3.1 Factoring-Based Complexity Assumptions

The RSA assumptions has been introduced in the seminal work by Rivest, Shamir and Adleman [90].

Definition 2.9 (RSA assumption (RSA)) *Let $l = l(\kappa)$ be a polynomial in the security parameter. Let $p, q \xleftarrow{\$} \mathbb{P}_{l,l+1}$ and $n = pq$. Let $\alpha \xleftarrow{\$} \{x | x < \phi(n), \gcd(x, \phi(n)) = 1\} \cap \mathbb{P}$ and $u \xleftarrow{\$} \mathbb{Z}_n^*$. We say that algorithm $\mathcal{A}$ $(t_{RSA}, \epsilon_{RSA})$-solves the RSA problem when, in time t_{RSA}, $\mathcal{A}$ has success probability at least ϵ_{RSA} in breaking the RSA problem such that*

$$\Pr\left[(x) \leftarrow \mathcal{A}(n, u, \alpha),\ x \in \mathbb{Z}_n^*,\ x^\alpha = u \bmod n\right] \leq \epsilon_{RSA},$$

where the probability is over the random choices of u, p, q, α and the random coins of $\mathcal{A}$.

Definition 2.10 *We say that the $(t_{RSA}, \epsilon_{RSA})$-RSA assumption holds, if no attacker can $(t_{RSA}, \epsilon_{RSA})$-solve the RSA problem.*

In 1997, Barić and Pfitzmann introduced the strong SRSA assumption [3].

Definition 2.11 (Strong RSA assumption (SRSA)) *Let $l = l(\kappa)$ be a polynomial in the security parameter. Let $p, q \xleftarrow{\$} \mathbb{P}_{l,l+1}$, and $n = pq$. Let $u \xleftarrow{\$} \mathbb{Z}_n^*$. The Strong RSA problem is to compute $(u^{1/e} \bmod n, e) \in \mathbb{Z}_n^* \times \mathbb{N}$ for some $e \geq 1$. We say that algorithm $\mathcal{A}$ $(t_{SRSA}, \epsilon_{SRSA})$-solves the SRSA problem when, in time t_{SRSA}, $\mathcal{A}$ has success probability at least ϵ_{SRSA} in breaking the SRSA problem such that*

$$\Pr\left[(x, y) \leftarrow \mathcal{A}(n, u),\ x \in \mathbb{Z}_n^*,\ y > 1,\ x^y = u \bmod n\right] \geq \epsilon_{SRSA},$$

where the probability is over the random choices of u, p, q and the random coins of $\mathcal{A}$.

Definition 2.12 *We say that the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds, if no attacker can $(t_{SRSA}, \epsilon_{SRSA})$-solve the SRSA problem.*

2.3.2 Discrete Logarithm Based Assumption

The discrete logarithm (DL) assumption is one of the most popular complexity assumptions used in cryptography. However, seldom cryptographic schemes solely rely on the DL assumption. Often, schemes are based on assumptions which imply the DL assumption.

Definition 2.13 (Discrete Logarithm problem) *Let $l = l(\kappa)$ be a polynomial. Let $\mathbb{G}$ be a group of prime order p with $|p|_2 = l$. Let $g \xleftarrow{\$} \mathbb{G}$ be a random generator of $\mathbb{G}$. The discrete logarithm problem (DL) in $\mathbb{G}$ is, given $g, g^x \in \mathbb{G}$ with $x \xleftarrow{\$} \mathbb{Z}_p$, to compute x. We say that algorithm $\mathcal{A}$ (t_{DL}, ϵ_{DL})-solves the DL problem in $\mathbb{G}$ when, in time t_{DL}, $\mathcal{A}$ has success probability at least ϵ_{DL} in breaking the DL problem such that*

$$\Pr\left[x \leftarrow \mathcal{A}\left(g, g^x\right)\right] \geq \epsilon_{DL},$$

where the probability is over the random choices of $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$ and the random coin tosses of $\mathcal{A}$.

Definition 2.14 *We say that the (t_{DL}, ϵ_{DL})-DL assumption holds, if no attacker can (t_{DL}, ϵ_{DL})-solve the DL problem.*

The computational Diffie-Hellman assumptions was introduced in the seminal work of Diffie and Hellman in 1976 [46].

Definition 2.15 (Computational Diffie-Hellman problem) *Let $l = l(\kappa)$ be a polynomial. Let $\mathbb{G}$ be a group of prime order p with $|p|_2 = l$. Let $g \overset{\$}{\leftarrow} \mathbb{G}$ be a random generator of $\mathbb{G}$. The computational Diffie-Hellman problem (CDH) in $\mathbb{G}$ is, given $g, g^a, g^b \in \mathbb{G}$ with $a, b \overset{\$}{\leftarrow} \mathbb{Z}_p$, to compute $g^{ab} \in \mathbb{G}$. We say that algorithm $\mathcal{A}$ $(t_{CDH}, \epsilon_{CDH})$-solves the CDH problem in $\mathbb{G}$ when, in time t_{CDH}, $\mathcal{A}$ has success probability at least ϵ_{CDH} in breaking the CDH problem such that*

$$\Pr\left[g^{ab} \leftarrow \mathcal{A}\left(g, g^a, g^b\right)\right] \geq \epsilon_{CDH},$$

where the probability is over g, a, b and the random coin tosses of $\mathcal{A}$.

Definition 2.16 *We say that the $(t_{CDH}, \epsilon_{CDH})$-CDH assumption holds, if no attacker can $(t_{CDH}, \epsilon_{CDH})$-solve the CDH problem.*

The Strong Diffie-Hellman (SDH) was introduced in 2004 by Boneh and Boyen [17]. Since then, several very efficient cryptographic schemes have been proposed that rely on SDH. Among them we can also find the shortest signature scheme that is secure in the standard model[64].

Definition 2.17 (Strong Diffie-Hellman assumption (SDH)) *Let $l = l(\kappa)$ be a polynomial. Let $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ be a bilinear group with $g_1 \overset{\$}{\leftarrow} \mathbb{G}_1$, $g_2 \overset{\$}{\leftarrow} \mathbb{G}_2$, and $|p|_2 = l$. Let $x \overset{\$}{\leftarrow} \mathbb{Z}_p$. We say that the $(q_{SDH}, \epsilon_{SDH}, t_{SDH})$-SDH assumption holds if for all t_{SDH}-time attackers $\mathcal{A}$ that are given $T = \left(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x\right) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ with $(q = q_{SDH})$ and $x \in \mathbb{Z}_p$ it holds that*

$$\Pr\left[(g_1^{1/(x+c)}, c) \leftarrow \mathcal{A}(T)\right] \leq \epsilon_{SDH},$$

where $c \in \mathbb{Z}_p$ and the probability is over the random choices for g_1, g_2, x and the random bits of $\mathcal{A}$.

2.4 Hash Functions

In the standard definition, a hash function $\mathsf{HF} = (\mathsf{HF.KeyGen}, \mathsf{HF.Eval})$ consists of two algorithms.

- $\mathsf{HF.KeyGen}(1^\kappa)$: given the security parameter κ this probabilistic algorithm outputs a hash key k_{HF} in the key space $\mathcal{K}_{\mathsf{HF}}$.
- $\mathsf{HF.Eval}(k_{\mathsf{HF}}, m)$: given a key $k_{\mathsf{HF}} \in \mathcal{K}_{\mathsf{HF}}$ and a message m from the message space $\mathcal{M}_{\mathsf{HF}}$ (usually we have $\mathcal{M}_{\mathsf{HF}} = \{0,1\}^*$) this deterministic algorithm outputs the hash value y in the hash space $\mathcal{Y}_{\mathsf{HF}}$.

2.4.1 Security of Hash Functions

Security of hash functions is defined through the following attack games between a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$.

CR – Fully Collision-Resistant Hash Functions. This is the standard security definition for cryptographically secure hash functions.

Hash key. The challenger $\mathcal{C}$ runs $\mathsf{HF.KeyGen}(1^\kappa)$ to obtain a fresh hash key k_{HF}. Next, $\mathcal{C}$ gives k_{HF} to $\mathcal{A}$.

Collision. Finally, $\mathcal{A}$ outputs $m, m' \in \mathcal{M}_{\mathsf{HF}}$ with $m \neq m'$.

TCR – Target Collision-Resistant Hash Functions. A weaker but also useful notion of security defines target collision-resistant hash functions. This primitive was first introduced in [82]. The security definition of target collision-resistant hash functions considers a slightly less powerful attacker. In particular, the attacker must choose m before it receives the hash function's key.

First message. In the first step, the adversary $\mathcal{A}$ outputs a message $m \in \mathcal{M}_{\mathsf{HF}}$.

Hash key. The challenger $\mathcal{C}$ runs $\mathsf{HF.KeyGen}(1^\kappa)$ to obtain a fresh hash key k_{HF}. Next, $\mathcal{C}$ gives k_{HF} to $\mathcal{A}$.

Collision. Finally, $\mathcal{A}$ outputs $m' \in \mathcal{M}_{\mathsf{HF}}$ with $m' \neq m$.

In each of the above games attacker $\mathcal{A}$ wins if m and m' collide, i.e. $\mathsf{HF.Eval}(k_{\mathsf{HF}}, m) = \mathsf{HF.Eval}(k_{\mathsf{HF}}, m')$. We denote the success probability of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win one of the above security games X with $X \in \{\mathsf{CR}, \mathsf{TCR}\}$ as $Adv_{\mathsf{HF},\mathcal{A},X}$.

Definition 2.18 (Secure Hash Function) *An adversary $\mathcal{A}$ is said to $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$-break the (full/target) collision-resistance (security) of hash function HF if $\mathcal{A}$ has success probability $Adv_{\mathsf{HF},\mathcal{A},X} = \epsilon$ with $X \in \{\mathsf{CR}, \mathsf{TCR}\}$ while running in time t_{HF}. HF is said to be $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$-secure if there exists no PPT adversary that $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$-breaks the (full/target) collision-resistance (security) of* HF *.*

Both types of collision-resistant hash functions can be used for message space extensions (from $\{0,1\}^l$ to $\{0,1\}^*$) for signature schemes. Using fully collision-resistant hash functions, messages longer than l bits simply have to be hashed before signing. However, the application of target collision-resistant hash function requires that a new k_{HF} has to be chosen for each new signature and prepended to the signature [10, 78].

Despite their common usefulness for domain extension, these primitives are fundamentally different. The general transformation from one-way functions to digital signature schemes [5, 82, 92] also shows that target collision-resistant hash functions can be constructed from one-way functions. In contrast, Simon showed that fully collision-resistant hash functions cannot be based on one-way function [99] using black-box constructions.

In the following, if we require a hash function to be simply 'collision-resistant' without specifying the type of collision-resistance, we always refer to a fully collision-resistant hash function. In signature schemes, the key of a fully collision-resistant hash function k_{HF} is usually generated once and then fixed in the schemes public key. If k_{HF} is clear from the context we may write $h(\cdot)$ for $\mathsf{HF.Eval}(k_{\mathsf{HF}}, \cdot)$.

2.5 Chaining Function

We will now define chaining functions $w : \mathcal{R}_{\text{chain}} \times \mathcal{M}_{\text{chain}} \rightarrow \mathcal{R}_{\text{chain}}$ that combine an input message $m \in \mathcal{R}_{\text{chain}}$ with a random value $r \in \mathcal{R}_{\text{chain}}$. Chaining Functions will be used in Chapter 3 to describe the signature schemes of Section 3.1 and Section 3.3. In the context of signature schemes, chaining functions have already been introduced by Cramer, Damgård, Pedersen in [44] (although not under a dedicated name). The abstraction presented here is very convenient in the signature description and for proving Lemma 3.3. In their SRSA-based signature scheme Naccache, Pointcheval, Stern instantiated the transformation of [44] while concentratating on a restricted class of chaining functions that are based on the XOR operation [80].

Definition 2.19 (Chaining Functions) *Let $l_{chain} = l_{chain}(\kappa)$ be a polynomial. Let $\mathcal{W}_\kappa$ for $\kappa \in \mathbb{N}$ be a collection of functions of the form $w : \mathcal{R}_{chain} \times \mathcal{M}_{chain} \rightarrow \mathcal{R}_{chain}$ with $|\mathcal{R}_{chain}| \leq 2^{l_{chain}}$. Let $\mathcal{W} = \{\mathcal{W}_\kappa\}_{\kappa\in\mathbb{N}}$. We say that $\mathcal{W}$ is $(t_{chain}, \epsilon_{chain})$-chaining if for all attackers $\mathcal{A}$ there exists a negligible function $\epsilon_{chain} = \epsilon_{chain}(\kappa)$ and the following properties hold for $w \overset{\$}{\leftarrow} \mathcal{W}_\kappa$:*

1. *For all $m \in \mathcal{M}_{chain}$ it holds that $w(\mathcal{R}_{chain}, m) = \mathcal{R}_{chain}$ (injectivity).*

2. *Given $z \in \mathcal{R}_{chain}$, $m \in \mathcal{M}_{chain}$, it is easy to find $r \in \mathcal{R}_{chain}$ such that $w(r, m) = z$ (invertibility).*

3. *For all $\bar{r} \in \mathcal{R}_{chain}$, it holds for all t_{chain}-time attackers $\mathcal{A}$ that*

$$\Pr\left[w(\bar{r}, m) = w(\bar{r}, m');\ (m, m') \leftarrow \mathcal{A}(\bar{r}, w),\ m, m' \in \mathcal{M}_{chain},\ m \neq m'\right] \leq \epsilon_{chain},$$

   where the probability is over the random coins of $\mathcal{A}$ (collision-resistance).[1]

4. *For all $m \in \mathcal{M}_{chain}$ and given* random *$r, r' \in \mathcal{R}_{chain}$, it holds for all t_{chain}-time attackers $\mathcal{A}$ that*

$$\Pr\left[w(r, m) = w(r', m');\ m' \leftarrow \mathcal{A}(r, r', m, w),\ m' \in \mathcal{M}_{chain}\right] \leq \epsilon_{chain},$$

   where the probability is computed over the random coins of $\mathcal{A}$ and the random choices of r and r' (second pre-image resistance).

[1]If w is also a permutation or just injective in the second input parameter, this requirement is trivially fulfilled.

If used in signature schemes, w is fixed at random during the key generation phase.

In [80], Naccache *et al.* implicitly show that $\mathcal{W} = \{\mathcal{W}_{2\kappa}\}_{\kappa\in\mathbb{N}}$ is chaining for $\mathcal{M}_{\text{chain}} = \{0,1\}^{\kappa}$ and $\mathcal{R}_{\text{chain}} = \{0,1\}^{2\kappa}$ with $\kappa \in \mathbb{N}$ if

$$\mathcal{W}_{2\kappa} = \{w(r,m) = r \oplus (m||m)\}.$$

The advantage of this function is its statistical security. The drawback is its inefficiency when signing long messages.

Lemma 2.20 *The above function $\mathcal{W}$ is $(\cdot, 2^{-\kappa})$-chaining according to Definition 2.19.*

Proof. Properties 1 and 2 are obvious. Property 3 follows from the fact that for any two $m, m' \in \{0,1\}^{\kappa}$ with $m \neq m'$ it also holds that $m||m \neq m'||m'$ and hence $r \oplus m||m \neq r \oplus m'||m'$ for all $r \in \mathcal{R}$. To show Property 4, suppose we are given a random $m \in \{0,1\}^k$ and random $r, r' \in \{0,1\}^{2\kappa}$. Assume that there exists $m' \in \{0,1\}^k$ such that $r \oplus m||m = r' \oplus m'||m'$. We then must have that $r \oplus r'$ can be represented as $\bar{r}||\bar{r}$ for some $\bar{r} \in \{0,1\}^k$. For a given random pair r and r' this only happens with probability $2^{-\kappa}$. $\square$

2.6 Combining Function

In this section, we introduce a new family of functions called combining functions. We will subsequently use the concept of combining functions to generalize several signature schemes without random oracles in Chapter 4.

Definition 2.21 (Combining Functions) *Let $\mathcal{V}_k$ for $k \in \mathbb{N}$ be a collection of functions of the form $z : \mathcal{R} \times \mathcal{M} \to \mathcal{Z}$ with $|\mathcal{Z}| \leq 2^k$. Let $\mathcal{V} = \{\mathcal{V}_k\}_{k\in\mathbb{N}}$. We say that $\mathcal{V}$ is $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining if for all attackers $\mathcal{A}$ there exist negligible functions $\epsilon_{comb}(k)$ and $\delta_{comb}(k)$ and the following properties hold for $z \overset{\$}{\leftarrow} \mathcal{V}_k$.*

1. *for all $m \in \mathcal{M}$ it holds that $|\mathcal{R}| = |\mathcal{Z}_m|$ where $\mathcal{Z}_m$ is defined as $\mathcal{Z}_m = z(\mathcal{R}, m)$. For all $m \in \mathcal{M}$ and all $t \in \mathcal{Z}$ there exists an efficient algorithm $z^{-1}(t,m)$ that, if $t \in \mathcal{Z}_m$, outputs the unique value $r \in \mathcal{R}$ such that $z(r,m) = t$, and $\perp$ otherwise.*

2. *for $t \overset{\$}{\leftarrow} \mathcal{Z}$ and $r' \overset{\$}{\leftarrow} \mathcal{R}$ we have for the maximal (over all $m \in \mathcal{M}$) statistical distance between r' and $z^{-1}(t,m)$ that*

$$\max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left|\Pr[r' = r] - \Pr[z^{-1}(t,m) = r]\right| \right\} \leq \delta_{comb}.$$

3. *for all $r \in \mathcal{R}$, it holds for all t_{comb}-time attackers $\mathcal{A}$ that*

$$\Pr\left[z(r,m) = z(r,m');\ (m,m') \leftarrow \mathcal{A}(z,r),\ m,m' \in \mathcal{M},\ m \neq m'\right] \leq \epsilon_{comb},$$

where the probability is taken over the random bits of $\mathcal{A}$.

Table 2.1: Examples of statistically secure combining functions. Let $\mathcal{V} = \{\mathcal{V}_k\}_{k\in\mathbb{N}}$ with $\mathcal{V}_k = \{z(r,m)\}$, $l, l_r, l_m \in \mathbb{N}$, $l_r > l_m$ and p be prime.

	$z(r,m)$	$\mathcal{R}$	$\mathcal{M}$	$\mathcal{Z}$	combining
EX1	$r + m \bmod p$	$\mathbb{Z}_p$	$\mathbb{Z}_p$	$\mathbb{Z}_p$	$(\cdot, 0, 0)$
EX2	$r \oplus m$	$\{0,1\}^l$	$\{0,1\}^l$	$\{0,1\}^l$	$(\cdot, 0, 0)$
EX3	$r + m$	$[0; 2^{l_r} - 1]$	$[0; 2^{l_m} - 1]$	$[0; 2^{l_r} + 2^{l_m} - 2]$	$(\cdot, 0, 2^{l_m - l_r})$

In the following, we assume that when used in signature schemes, $z \overset{\$}{\leftarrow} \mathcal{V}_k$ is chosen uniformly at random during the key generation phase.

In Table 2.1, we present three concrete examples (EX1, EX2, EX3) of statistically secure combining functions. The following lemma shows that these examples are valid combining functions with respect to Definition 2.21.

Lemma 2.22 EX1 *and* EX2 *constitute* $(\cdot, 0, 0)$*-combining functions and* EX3 *constitutes a* $(\cdot, 0, 2^{l_m - l_r})$*-combining function.*

Proof. Let us first analyze EX1 and EX2. We have that $\mathcal{M} = \mathcal{R} = \mathcal{Z} = \mathcal{Z}_m$ for all $m \in \mathcal{M}$ and we can efficiently compute r as $r = t - m \bmod p$ or $r = t \oplus m$ for *all* given $t \in \mathcal{Z}$ and $m \in \mathcal{M}$. Furthermore, since z is bijective in both input parameters $z^{-1}(t,m)$ is uniformly distributed in $\mathcal{R}$ for all $m \in \mathcal{M}$ and random $t \in \mathcal{Z}$. Thus, $\delta_{\text{comb}} = 0$. Finally, since z is a bijection in the second input parameter, it is collision-free (property 3) in both examples and we have that $\epsilon_{\text{comb}} = 0$. Now, let us analyze EX3. For given $m \in \mathcal{M}$ and $t \in \mathcal{Z}$, $z^{-1}(t,m)$ outputs $r = t - m$ if $t - m \in \mathcal{R}$ and $\perp$ otherwise. To show that z is collision-free, observe that $m \neq m'$ implies $r + m \neq r + m'$ for all $r \in \mathcal{R}$. To analyze $D = \max_{m\in\mathcal{M}} \left\{1/2 \cdot \sum_{r\in\mathcal{R}} |\Pr[r' = r] - \Pr[z^{-1}(t,m) = r]|\right\}$ first note that for $t' \overset{\$}{\leftarrow} \mathcal{Z}_m$, $z^{-1}(t', m)$ is uniform in $\mathcal{R}$ since $|\mathcal{Z}_m| = |\mathcal{R}|$ implies that $z^{-1}(\cdot, m)$ defines a bijection from $\mathcal{Z}_m$ to $\mathcal{R}$. For $t' \overset{\$}{\leftarrow} \mathcal{Z}_m$ and $t \overset{\$}{\leftarrow} \mathcal{Z}$ we get

$$\begin{aligned}
D &= \max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left|\Pr[r' = r] - \Pr[z^{-1}(t,m') = r]\right| \right\} \\
&= \max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left|\Pr[z^{-1}(t',m) = r] - \Pr[z^{-1}(t,m) = r]\right| \right\} \\
&\leq \max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{t_0\in\mathcal{Z}_m} |\Pr[t' = t_0] - \Pr[t = t_0]| \right\} \\
&= \frac{|\mathcal{Z}_m|}{2} \cdot \left(\frac{1}{|\mathcal{Z}_m|} - \frac{1}{|\mathcal{Z}|}\right) = \frac{|\mathcal{Z}| - |\mathcal{Z}_m|}{2|\mathcal{Z}|} \\
&= \frac{2^{l_m} - 2}{2(2^{l_m} + 2^{l_r} - 2)} \\
&\leq 2^{l_m - l_r}
\end{aligned}$$

□

Three further examples of combining functions can be obtained when first applying a (t_h, ϵ_h)-collision-resistant hash function that maps (long) messages to $\mathcal{M}$. Lemma 2.23 guarantees that the results are still combining according to Definition 2.21.

Lemma 2.23 *Let $\mathcal{Z}_{comb}$ be a $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining function with message space $\mathcal{M}_{comb}$, randomness space $\mathcal{R}_{comb}$, and output space $\mathcal{Y}_{comb}$. Let $\mathsf{HF} = (\mathsf{HF.KeyGen}, \mathsf{HF.Eval})$ be a $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$-collision-resistant hash function with message space $\mathcal{M}_{\mathsf{HF}}$ and hash space $\mathcal{Y}_{\mathsf{HF}}$. Then it holds that $\mathcal{Z}'_{comb} = \{\mathcal{Z}'_\kappa\}_{\kappa\in\mathbb{N}}$ with $\mathcal{Z}'_\kappa = \{z(r, \mathsf{HF.Eval}(k_{\mathsf{HF}}, m)) | z \xleftarrow{\$} \mathcal{Z}_\kappa, k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}(1^\kappa)\}$ is $(\min\{t_{comb}, t_{\mathsf{HF}}\}, \epsilon_{comb} + \epsilon_{\mathsf{HF}}, \delta_{comb})$-combining with message space $\mathcal{M}_{\mathsf{HF}}$, randomness space $\mathcal{R}_{comb}$, and output space $\mathcal{Y}_{comb}$.*

Proof. Assume $\mathcal{V}$ is $(t_{\text{comb}}, \epsilon_{\text{comb}}, \delta_{\text{comb}})$-combining and $\mathcal{H}$ is (t_h, ϵ_h)-collision-resistant. Let $z \xleftarrow{\$} \mathcal{V}_k$ and $h \xleftarrow{\$} \mathcal{H}_{l_m}$. Set $\mathcal{M}' = \{0,1\}^*$ and $z'(r, m) = z(r, h(m))$. First observe that given $m \in \mathcal{M}'$ and $t \in \mathcal{Z}_m$ we can always compute $r \in \mathcal{R}$ (or $\perp$) just by finding an appropriate r for $h(m), z(r, h(m))$ using the properties of z.

Now, for contradiction assume that an attacker $\mathcal{A}$ can find collisions for z' in time $\min\{t_{\text{comb}}, t_h\}$ with probability better than $\epsilon_h + \epsilon_{\text{comb}}$. Let (m, m') be such a collision. Then, we have either found a collision (m, m') of the hash function (if $h(m) = h(m')$) or a collision $(h(m), h(m'))$ in the combining function $(h(m) \neq h(m'))$. In the first case, $\mathcal{A}$ has computed a hash collision in time equal or less t_h with a probability greater than $\epsilon_h + \epsilon_{\text{comb}} \geq \epsilon_h$. This contradicts the fact that $\mathcal{H}$ is (t_h, ϵ_h)-collision-resistant. On the other hand, if $\mathcal{A}$ has found a collision in the combining function, this means that $\mathcal{A}$ can break the collision-resistance of $\mathcal{V}$ in time less or equal than t_{comb} with probability greater than $\epsilon_h + \epsilon_{\text{comb}} \geq \epsilon_{\text{comb}}$. This contradicts the fact that $\mathcal{V}$ is $(t_{\text{comb}}, \epsilon_{\text{comb}}, \delta_{\text{comb}})$-combining.

To proof the probability bound δ_{comb}, observe that since $h(\mathcal{M}') \subseteq \mathcal{M}$ we finally have

$$\begin{aligned} & \max_{m\in\mathcal{M}'} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left| \Pr[r' = r] - \Pr[z^{-1}(t, h(m)) = r] \right| \right\} \\ \leq \; & \max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left| \Pr[r' = r] - \Pr[z^{-1}(t, m) = r] \right| \right\} \\ \leq \; & \delta_{\text{comb}}. \end{aligned}$$

□

2.7 Pseudo-Random Function

A collision-resistant hash function $\mathsf{PRF} = (\mathsf{PRF.KeyGen}, \mathsf{PRF.Eval})$ consists of two algorithms.

- $\mathsf{PRF.KeyGen}(1^\kappa)$: given the security parameter κ this probabilistic algorithm outputs a hash key k_{PRF} in the key space $\mathcal{K}_{\mathsf{PRF}}$.

- $\mathsf{PRF.Eval}(k_{\mathsf{PRF}}, m)$: given a key $k_{\mathsf{PRF}} \in \mathcal{K}_{\mathsf{PRF}}$ and a message m from the message space $\mathcal{M}_{\mathsf{PRF}}$ this deterministic algorithm outputs the value y in the output space $\mathcal{Y}_{\mathsf{PRF}}$.

Security of Pseudo-Random Functions. Security of a pseudo-random function is defined through the following attack game between a challenger $\mathcal{C}$ and attacker $\mathcal{A}$.

Key generation. In the first step, $\mathcal{C}$ generates a fresh key $k_{\mathsf{PRF}} \leftarrow \mathsf{PRF.KeyGen}(1^\kappa)$.

Adaptive queries. Next, $\mathcal{A}$ can adaptively query oracle $O(k_{\mathsf{PRF}}, \cdot)$ with messages $m_1, \ldots, m_q$ for $q = q(\kappa)$. On input $m_i \in \mathcal{M}_{\mathsf{PRF}}$ with $i \in [1; q]$, $O(k_{\mathsf{PRF}}, \cdot)$ outputs $\mathsf{PRF.Eval}(k_{\mathsf{PRF}}, m_i) \in \mathcal{Y}_{\mathsf{PRF}}$.

Test Query. Finally, $\mathcal{A}$ outputs a test query $m^* \in \mathcal{M}_{\mathsf{PRF}}$ with $m^* \notin \{m_1, \ldots, m_q\}$.

Challenge. The challenger draws a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ computes $y_0^* \leftarrow \mathsf{PRF.Eval}(k_{\mathsf{PRF}}, m^*)$. If $b = 1$, $\mathcal{C}$ randomly draws $y_1^* \in \mathcal{Y}_{\mathsf{PRF}}$. Then it sends y_b^* to $\mathcal{A}$.

Attacker's response. Given y_b^*, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ indicating its guess of b.

We denote the advantage of $\mathcal{A}$ to win the above security game as

$$Adv_{\mathcal{A},\mathsf{PRF}} = |\Pr[b = b'] - 1/2|.$$

Definition 2.24 (Pseudo-random function) *We say that $\mathcal{A}$ $(t_{\mathsf{PRF}}, \epsilon_{\mathsf{PRF}})$-breaks the security of* PRF *if $\mathcal{A}$ runs in time t_{PRF} and has success probability the $Adv_{\mathcal{A},\mathsf{PRF}} \geq \epsilon_{\mathsf{PRF}}$ to win in the above security game.* PRF *is called $(t_{\mathsf{PRF}}, \epsilon_{\mathsf{PRF}})$-secure if there exists no polynomial-time attacker $\mathcal{A}$ that $(t_{\mathsf{PRF}}, \epsilon_{\mathsf{PRF}})$-breaks the security of* PRF .

2.8 Chameleon Hash Function

Chameleon hash functions were (explicitly) introduced in 2000 [70]. Intuitively, a chameleon hash function is a randomized hash function for which it is very hard to find collisions. However given its secret key, one can easily *compute* collisions even for *given messages*. Formally, a chameleon hash function $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ consists of three efficient algorithms.

- $\mathsf{CH.Gen}(1^\kappa)$: this probabilistic algorithm outputs a secret key SK_{CH} and a public key PK_{CH}.

- $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r)$: given a public key PK_{CH}, a message m from the message space $\mathcal{M}_{\mathsf{CH}}$, and a random r from the randomization space $\mathcal{R}_{\mathsf{CH}}$ this deterministic algorithm $\mathsf{CH.Eval}$ outputs a chameleon hash value y in the hash space $\mathcal{Y}_{\mathsf{CH}}$.

- $\mathsf{CH.Coll}(SK_{\mathsf{CH}}, r, m, m')$: this algorithm deterministically outputs, on input SK_{CH} and a triple $(r, m, m') \in \mathcal{R}_{\mathsf{CH}} \times \mathcal{M}_{\mathsf{CH}} \times \mathcal{M}_{\mathsf{CH}}$, a value $r' \in \mathcal{R}_{\mathsf{CH}}$ such that $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) = \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m', r')$.

An *invertible* chameleon hash function $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll}, \mathsf{CH.Inv})$ [96] additionally consists of a fourth algorithm $\mathsf{CH.Inv}$:

- $\mathsf{CH.Inv}(SK_{\mathsf{CH}}, y, m)$: given the secret key SK_{CH}, a hash value $y \in \mathcal{Y}_{\mathsf{CH}}$ and a message $m \in \mathcal{M}_{\mathsf{CH}}$ this algorithm outputs $r \in \mathcal{R}_{\mathsf{CH}}$ with $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) = y$.

Definition 2.25 (Collision-Resistant Chameleon Hash Function) *We say that* CH *is* $(t_{\mathsf{CH}}, \epsilon_{\mathsf{CH}})$*-secure if for all* t_{CH}*-time adversaries* $\mathcal{A}$ *that are only given* PK_{CH} *it holds that*

$$\Pr\left[\begin{array}{c}\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) = \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m', r');\\ (SK_{\mathsf{CH}}, PK_{\mathsf{CH}}) \leftarrow \mathsf{CH.Gen}(1^{\kappa}),\ (m, m', r, r') \leftarrow \mathcal{A}(PK_{\mathsf{CH}}),\\ r, r' \in \mathcal{R}_{\mathsf{CH}},\ m, m' \in \mathcal{M}_{\mathsf{CH}},\ m \neq m'\end{array}\right] \leq \epsilon_{\mathsf{CH}},$$

where the probability is over the random choices of PK_{CH} *and the coin tosses of* $\mathcal{A}$*. We say that* CH *is strongly collision-resistant if we solely require that* $(m', r') \neq (m, r)$ *(rather than* $m \neq m'$*).*

We also require that for arbitrary but fixed keys $(PK_{\mathsf{CH}}, SK_{\mathsf{CH}})$ output by CH.Gen and for random $r \in \mathcal{R}_{\mathsf{CH}}$, $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r)$ is distributed equally for all messages $m \in \mathcal{M}_{\mathsf{CH}}$. In the case of invertible chameleon hash functions we additionally need that for randomly distributed values $y \in \mathcal{Y}_{\mathsf{CH}}$ and all $m \in \mathcal{M}_{\mathsf{CH}}$ the distribution of $\mathsf{CH.Inv}(SK_{\mathsf{CH}}, y, m)$ is indistinguishable from uniform in $\mathcal{R}_{\mathsf{CH}}$.

The security of chameleon hash functions can be based on standard assumptions like the discrete logarithm assumption [70] or the factoring assumption [70]. We remark that most of the existing chameleon hash functions are actually *strongly* secure. To prove this, all we need to show is that for all PK_{CH} and all m, $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r)$ is injective in r. Usually, this can be done simply by inspection.

Lemma 2.26 *Let* CH *be a* $(t_{\mathsf{CH}}, \epsilon_{\mathsf{CH}})$*-secure chameleon hash function. If for all messages* $m \in \mathcal{M}_{\mathsf{CH}}$ *and all* PK_{CH} *generated using* $\mathsf{CH.Gen}(1^{\kappa})$*,* $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r)$ *is injective in* r*, then* CH *is strongly secure.*

PROOF. To prove Lemma 2.26 observe that if $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r)$ is an injection for all m and all PK_{CH}, we always have for r, r' with $r \neq r'$ that $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) \neq \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r')$. Therefore, there cannot exist values r, r' with $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) = \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r')$. This implies that for all collisions $(m, r), (m', r')$ with $(m, r) \neq (m', r')$ we always must have that $m \neq m'$. But this is the standard definition of security. $\square$

2.9 Multi-Generator Programmable Hash Function in Prime Order Groups

In our (ring) signature scheme (Chapter 6) we use the multi-generator programmable hash function by Hofheinz and Kiltz in groups with known prime order [64] which in turn is derived from the CDH-based signature scheme by Waters [102]. Intuitively, the multi-generator programmable hash function offers two indistinguishable key generation routines. The first samples the elements of the key at random. A given input message is evaluated by multiplying a subset of these elements together. The second key generation routine also outputs a secret key. This key allows to evaluate the multi-generator programmable hash function in a different but very useful way. Formally, a multi-generator programmable hash function $\mathsf{SIG} = (\mathsf{PHF.KeyGen}, \mathsf{PHF.Eval}, \mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$ consists of four algorithms.

2.9.1 Multi-Generator Programmable Hash Function

Definition 2.27 (Multi-Generator PHF) *The multi-generator programmable hash function consists of four algorithms.*

- $\mathsf{PHF.KeyGen}(1^\kappa, l)$*: given* 1^κ, $l = l(\kappa)$ *and a group* $\mathbb{G}$ *of prime order* p *with* $|p|_2 = l_p = l_p(\kappa)$, $\mathsf{PHF.KeyGen}$ *returns* $l+1$ *random group generators* $u_0, u_1, \ldots, u_l \in \mathbb{G}$.
- $\mathsf{PHF.Eval}(u_0, u_1, \ldots, u_l, m)$*: Given the* u_i *and a message* $m \in \{0,1\}^l$, $\mathsf{PHF.Eval}$ *outputs*

$$u(m) = u_0 \prod_{i=1}^{l} u_i^{m_i},$$

  where $(m_l, \ldots, m_1)$ *is the binary representation of* m: $m = \sum_{i=1}^{l} m_i 2^{i-1}$.
  The pair $(\mathsf{PHF.KeyGen}, \mathsf{PHF.Eval})$ *is called a group hash function.*
- $\mathsf{PHF.TrapGen}(1^\kappa)$*: on input* 1^κ, l *and generators* $g, h \in \mathbb{G}$, *the algorithm* $\mathsf{PHF.TrapGen}$ *randomly chooses* $a'_0, a_1, \ldots, a_l \in \{-1, 0, 1\}$ *and* $b_0, b_1, \ldots, b_l \in \mathbb{Z}_p$. *Next, it sets* $a_0 = a'_0 - 1$ *and outputs* $l+1$ *group elements* $u_i = g^{a_i} h^{b_i}$ *for* $i = 0, 1, \ldots, l$ *and the trapdoor* $(a_0, a_1, \ldots, a_l, b_0, b_1, \ldots, b_l)$.
- $\mathsf{PHF.TrapEval}(a_0, a_1, \ldots, a_l, b_0, b_1, \ldots, b_l, m)$*: Now, given* $(a_0, a_1, \ldots, a_l, b_0, b_1, \ldots, b_l)$ *and a message* m, *the algorithm* $\mathsf{PHF.TrapEval}$ *outputs* $a(m) = a_0 + \sum_{i=1}^{l} a_i m_i$ *and* $b(m) = b_0 + \sum_{i=1}^{l} b_i m_i$. *Note that when the* u_i *have been computed by* $\mathsf{PHF.TrapGen}$ *it clearly holds that* $u(m) = u_0 \prod_{i=1}^{l} u_i^{m_i} = g^{a(m)} h^{b(m)}$.

Hofheinz and Kiltz showed that for every fixed polynomial $q = q(\kappa)$ the multi-generator programmable hash function is $(1, q, 0, P_{q,l})$-programmable where $P_{q,l} = \mathcal{O}\left(\frac{1}{q\sqrt{l}}\right)$. This means, that the group elements output by $\mathsf{PHF.KeyGen}$ and $\mathsf{PHF.Eval}$ are *equally* distributed. Furthermore it holds for all possible input parameters to $\mathsf{PHF.TrapGen}$ and all $M_1, \ldots, M_{q+1} \in \{0,1\}^l$ with $M_{q+1} \neq M_i$ for $i \leq q$ that

$$\Pr[a(M_{q+1}) = 0 \ \wedge \ a(M_1), \ldots, a(M_q) \neq 0] \geq P_{q,l}.$$

The corresponding proof and further details on programmable hash functions can be found in the original paper [64]. A similar but (asymptotically) weaker result ($P_{q,l} = \frac{1}{8(l+1)q}$) was implicitly given by Waters in [102].

2.10 Signature Scheme

A digital signature scheme SIG=($\mathsf{SIG.KeyGen}$,$\mathsf{SIG.Sign}$,$\mathsf{SIG.Verify}$) consists of three algorithms.

- $\mathsf{SIG.KeyGen}(1^\kappa)$: this probabilistic polynomial time algorithm outputs, given the security parameter, a secret key SK and a public key PK.
- $\mathsf{SIG.Sign}(SK, m)$: given SK and a message m from the message space $\mathcal{M}_{\mathsf{SIG}}$, this algorithm outputs a signature σ in the signature space $\mathcal{Y}_{\mathsf{SIG}}$ on m.
- $\mathsf{SIG.Verify}(PK, m, \sigma)$: given the public key PK, a message $m \in \mathcal{M}_{\mathsf{SIG}}$, and a purported signature $\sigma \in \mathcal{Y}_{\mathsf{SIG}}$, this algorithm check whether σ is a legitimate signature on m signed by the holder of the secret key corresponding to PK. On successful verification the algorithm outputs 1, otherwise 0.

In the rest of this work, we only consider correct signature schemes.

Definition 2.28 (Correctness) *A signature scheme* SIG*=(*$\mathsf{SIG.KeyGen}$*,*$\mathsf{SIG.Sign}$*,*$\mathsf{SIG.Verify}$*) is called correct if for all* $m \in \mathcal{M}_{\mathsf{SIG}}$

$$\Pr\left[\mathsf{SIG.Verify}(PK, m, \mathsf{SIG.Sign}(SK, m)) = 1;\ (SK, PK) \leftarrow \mathsf{SIG.KeyGen}(1^\kappa)\right] = 1.$$

2.11 Security Notions

We restrict ourself to security definitions that have proven useful to obtain efficient hash-and-sign signature schemes. As usual in cryptography, security is formalized by first specifying a successful attacker (or forger) $\mathcal{A}$ against a signature scheme. Each attacker can be given (restricted) access to additional *attack capabilities*. Usually these capabilities are modeled as signing oracles that allow $\mathcal{A}$ to obtain sample signatures on (several) messages (of his choice). Furthermore we need to specify what it means for $\mathcal{A}$ to be successful. In the existing security games of signature schemes, $\mathcal{A}$ is successful if it can output a new signature σ^* that has not been obtained by a signing oracle before. The hardness of computing a forgery differs in $\mathcal{A}$'s degree of freedom to also specify the message m^* on which σ^* is a valid signature on. For example, m^* can be given to $\mathcal{A}$ at the very beginning of the security game (allowing no freedom of choice for $\mathcal{A}$ at all) or m^* can be freely chosen by $\mathcal{A}$ at the end of the attack game. We refer to the freedom of $\mathcal{A}$ in specify m^* as the (required) *success definition.* The combination of provided attack capabilities and success definition is called *the attack model.* Correspondingly, a signature scheme is called secure in a specific attack model if (under some complexity assumption) no polynomial-time attacker can exist that has access to the granted attack capabilities and outputs the required type of forgery.

Attack Capabilities. The attack capabilities are formalized by signing oracles $O(SK, \cdot)$ that allow $\mathcal{A}$ to obtain (several) message/signature pairs. These oracles can be very restrictive such that the attacker may not choose any of the signed messages (known/random message attacks (RMA)) at all or more powerful allowing $\mathcal{A}$ to specify (prior to receiving the scheme's public key and the signatures) the messages on which it wants to receive signatures on (generic chosen message attacks (gCMA)). The most powerful oracle allows $\mathcal{A}$ to adaptively query messages that even may depend on the scheme's public key and previously received signatures (adaptive chosen message attacks (aCMA)). These oracles constitute a hierarchy. Any attacker that is successful when only allowed to launch random message attacks is also successful when granted access to an oracle for generic chosen message attacks. Similarly, any attacker that is successful when allowed to launch generic chosen message attacks is also successful when granted access to an oracle for adaptive chosen message attacks.

Success Definitions. In the literature we can find three success definitions. Existential (message) forgeries allow $\mathcal{A}$ to output any message/signature pair (m^*, σ_*) such that m^* has not been queried before. Selective (message) forgeries require that $\mathcal{A}$ *must specify* m^* at the beginning of the attack game. Universal (message) forgeries require that m^* *is given* to $\mathcal{A}$ at the beginning to the attack game. These types of forgery are increasingly powerful. Any $\mathcal{A}$ that can compute universal forgeries of a signature scheme SIG can also compute selective forgeries. Similarly, any attacker that can compute selective forgeries can also compute existential forgeries. The corresponding security notions are called existential message unforgeability (EMUF), selective message unforgeability (SMUF), and universal message unforgeability (UMUF).

In the following we will focus on existential unforgeability as it provides the highest security guarantees.

2.11.1 Existential Message Unforgeability

Existential Message Unforgeability under Random Message Attacks (Random Security) – $\mathsf{EMUF\text{-}RMA}$. In this attack model, $\mathcal{A}$ is completely passive.

Signature queries, Public Key Generation and Signature Output. The challenger runs $\mathsf{SIG.KeyGen}(1^\kappa)$ to obtain (SK, PK). The adversary is given q random messages $m_1, \ldots, m_q \xleftarrow{\$} \mathcal{M}$, the public key PK, and q signatures $\sigma_1, \ldots, \sigma_q$ such that $\mathsf{SIG.Verify}(PK, m_i, \sigma_i) = 1$ for $i \in \{1, \ldots, q\}$.

Output. The attacker outputs (m^*, σ^*) with $m^* \notin \{m_1, \ldots, m_q\}$ and $\mathsf{SIG.Verify}(PK, m^*, \sigma^*) = 1$.

Existential Message Unforgeability under Generic Chosen Message Attacks (Weak Security) – $\mathsf{EMUF\text{-}gCMA}$. In this attack model, the attacker must specify the signature queries before it receives the public key.

Signature queries. At first the adversary sends a list of q signature queries $m_1, \ldots, m_q \in \mathcal{M}$ to the signing oracle $\mathcal{O}_{\mathsf{gCMA}}(SK, \cdot)$.

Public Key Generation and Signature Output. In the next phase, the public key PK is given to the adversary together with q signatures $\sigma_1, \ldots, \sigma_q$ s.t. $\mathsf{SIG.Verify}(PK, m_i, \sigma_i) = 1$ for $i \in \{1, \ldots, q\}$.

Output. The attacker outputs (m^*, σ^*) with $m^* \notin \{m_1, \ldots, m_q\}$ and $\mathsf{SIG.Verify}(PK, m^*, \sigma^*) =$

1.

Existential Message Unforgeability under Adaptive Chosen Message Attacks (Full Security) – EUF-aCMA. The standard notion of security for signature schemes is called existential unforgeability under adaptive chosen message attacks [62]. Here the adversary is given access to a signing oracle $\mathcal{O}_{\mathsf{aCMA}}(SK, \cdot)$ to adaptively query signatures.

Setup. In the setup phase, the public key PK is given to the adversary.

Signature queries. The adversary adaptively queries $\mathcal{O}_{\mathsf{aCMA}}(SK, \cdot)$ with q messages of his choice, $m_1, \ldots, m_q \in \mathcal{M}$, and obtains q signatures $\sigma_1, \ldots, \sigma_q$ with $\mathsf{SIG.Verify}(PK, m_i, \sigma_i) = 1$ for $i \in \{1, \ldots, q\}$.

Output. The attacker outputs (m^*, σ^*) with $m^* \notin \{m_1, \ldots, m_q\}$ and $\mathsf{SIG.Verify}(PK, m^*, \sigma^*) = 1$.

2.11.2 Definition of Security

We denote the success probability of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win in one of the above security games Y with $Y \in \{\mathsf{RMA}, \mathsf{gCMA}, \mathsf{aCMA}\}$ as $Adv_{\mathsf{SIG},\mathcal{A},Y}$.

Definition 2.29 (Secure Signature Scheme) *An adversary $\mathcal{A}$ is said to (q, t, ϵ)-break the existential message unforgeability of signature scheme* SIG *under random/generic chosen/adaptive chosen message attacks if $\mathcal{A}$ has success probability $Adv_{\mathsf{SIG},\mathcal{A},Y} = \epsilon$ with and $Y \in \{\mathsf{RMA}, \mathsf{gCMA}, \mathsf{aCMA}\}$ after generating at most q queries and running in time t.* SIG *is said to be (q, t, ϵ)-secure if there exists no PPT adversary that (q, t, ϵ)-breaks the existential unforgeability of* SIG*. In case $q = 1$,* SIG *is called a (t, ϵ)-secure one-time signature scheme. A signature scheme is called strongly secure if in the above security games $\mathcal{A}$ may also output a forgery with $m^* \in \{m_1, \ldots, m_q\}$ but $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$.*

2.12 Two-Tier Signature Schemes

Two-tier signature schemes were introduced by Bellare and Shoup [13]. Informally, a two-tier signature can be regarded as a 'one-time signature generator' that is identified by some long-term key material (SK, PK). The holder of the secret key SK can issue fresh one-time key pairs (osk, opk). However, in contrast to usual one-time signatures the one-time key material is related to each other – via the long-term secrets – and not totally independent. Therefore, signature generation and verification also require these values as input. As two-tier signature schemes have weaker security properties than classical signature schemes they allow for more efficient instantiations. A two-tier signature scheme $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$ consists of the following algorithms.

- $\mathsf{TT.KeyGen}(1^\kappa)$: outputs a secret key SK and a public key PK.

- $\mathsf{TT.OTKeyGen}(SK, PK)$: given (SK, PK) this algorithm outputs new one-time key material consisting of the one-time secret key osk and the one-time public key opk.
- $\mathsf{TT.Sign}(SK, osk, m)$: takes as input the secret keys SK and osk and the message $m \in \mathcal{M}_{\mathsf{TT}}$ and outputs a signature σ on m.
- $\mathsf{TT.Verify}(PK, opk, m, \sigma)$: if σ is a signature on message m with respect to public keys PK and opk this algorithm outputs 1, otherwise 0.

As with classical digital signatures, we subsequently only consider correct two-tier signature schemes.

Definition 2.30 (Correctness of two-tier signature schemes) *We call a two-tier signature scheme* TT=($\mathsf{TT.KeyGen}$,$\mathsf{TT.OTKeyGen}$,$\mathsf{TT.Sign}$,$\mathsf{TT.Verify}$) *correct if for all* $m \in \mathcal{M}_{\mathsf{TT}}$

$$\Pr\left[\begin{matrix}\mathsf{TT.Verify}(PK, opk, m, \mathsf{TT.Sign}(SK, osk, m)) = 1; \\ (SK, PK) \leftarrow \mathsf{TT.KeyGen}(1^\kappa); (osk, opk) \leftarrow \mathsf{TT.OTKeyGen}(SK, PK)\end{matrix}\right] = 1.$$

2.12.1 Security of Two-Tier Signature Schemes

The existing security definition is due to Bellare and Shoup [13].

We define security of two-tier signature schemes in the spirit of the security notions of classical signature schemes. The intuition behind the definitions is that we do not want the adversary to be able to break *any* of the issued one-time signature schemes (identified with the corresponding one-time key material). To model this the adversary is granted access to the one-time oracles of *each* one-time signature schemes.

Adaptively Secure Two-tier Signature Schemes. In the original security definition the adversary is *first* given access to the one-time public keys [13].

Public Key and Signature Generation. In the first phase the adversary is provided with a public key PK.

Signature queries. In the next step, the adversary is granted q-time access to the oracle $\mathcal{O}_{SK,\text{adapt}}(\cdot)$. On input index $i \in [1; q]$ this oracle outputs the one-time public key opk_i. Given an index $i \in [1; q]$ *and a* message m_i, $\mathcal{O}_{SK,\text{adapt}}(\cdot)$ outputs a one-time signature σ_i on m_i under one-time key opk_i such that $\mathsf{TT.Verify}(PK, opk_i, m_i, \sigma_i) = 1$. If opk_i has not been queried before, $\mathcal{O}_{SK,\text{adapt}}(\cdot)$ outputs both opk_i and σ_i. To implement the one-time functionality we require that the adversary can query $\mathcal{O}_{SK,\text{adapt}}(\cdot)$ for each i only once.

Output. The attacker outputs a forgery (m^*, σ^*) such that there exists $i \in [1; q]$ with $m^* \neq m_i$ and $\mathsf{TT.Verify}(PK, opk_i, m^*, \sigma^*) = 1$.

We denote the success probability of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win the adapt security game as $Adv_{\mathsf{TT},\mathcal{A},\text{adapt}}$ and the adaptive security game as $Adv_{\mathsf{TT},\mathcal{A},\text{adapt}}$.

Definition 2.31 (Secure Two-Tier Signature Scheme) *An adversary $\mathcal{A}$ (q,t,ϵ)-breaks the adaptive security of a given two-tier signature scheme* TT*if $\mathcal{A}$ has success probability $Adv_{\mathsf{TT},\mathcal{A},adapt} = \epsilon$ after generating at most q one-time queries and running in time t.* TT*is said to be adaptively (q,t,ϵ)-secure if there exists no PPT adversary that (q,t,ϵ)-breaks the adaptive security of* TT*. For convenience we call a two-tier signature scheme strongly secure if the generated one-time signature schemes are strongly secure (i.e. $(m^*,\sigma^*) \notin \{(m_1,\sigma_1),\ldots,(m_q,\sigma_q)\}$).*

2.13 Ring Signature Schemes

A ring signature scheme allows a signer to sign on behalf of a group of users, the so-called ring; the only condition is that the signer must also be part of this ring. Technically, a ring is represented by the set of public keys that correspond to the identities of the ring members. Using his private key, the signer can now sign a message such that anyone can check whether the signature has been generated by one of the ring members. At the same time, there exists no possibility to discover the actual signer. Ring signatures provide signer anonymity in a very strong sense. In contrast to group signature schemes [38], the anonymity of the signer cannot be revoked. What makes ring signature schemes very flexible is that no central management is needed and that the signer can freely choose the public keys in the ring even without their owners' consent. Direct applications for ring signature schemes include designated verifier signatures [69] and secret leaking [91], but ring signature schemes are in general useful in applications where signer anonymity is desired.

2.13.1 Definition

A ring signature scheme $\mathsf{RSIG} = (\mathsf{RSIG.KeyGen}, \mathsf{RSIG.Sign}, \mathsf{RSIG.Verify})$ consists of the following algorithms.

- $\mathsf{RSIG.KeyGen}(1^\kappa)$: generates a secret and public key (SK, PK).
- $\mathsf{RSIG.Sign}(R, SK_i, m)$: takes as input a tuple of public keys $R = (PK_1,\ldots,PK_n)$, a secret key SK_i with $i \in \{1,\ldots,n\}$ and a message $m \in \mathcal{M}_{\mathsf{RSIG}}$ and outputs a signature σ.
- $\mathsf{RSIG.Verify}(R, m, \sigma)$: processes R, a message m and a signature σ to check whether σ is a legitimate signature on m signed by a holder of a secret key corresponding to one of the public keys in R. Accordingly, the algorithm outputs 1 to indicate a successful verification and 0 otherwise.

We subsequently only consider correct ring signature schemes.

Definition 2.32 (Correctness of ring signature schemes) *We say that a ring signature scheme* $\mathsf{RSIG} = (\mathsf{RSIG.KeyGen}, \mathsf{RSIG.Sign}, \mathsf{RSIG.Verify})$ *is correct if for all $m \in \mathcal{M}_{\mathsf{RSIG}}$*

and all possible rings R and all i such that PK_i is a public key in R

$$\Pr\begin{bmatrix}\mathsf{RSIG.Verify}(R, m, \mathsf{RSIG.Sign}(R, SK_i, m)) = 1; \\ (SK, PK) \leftarrow \mathsf{RSIG.KeyGen}(1^\kappa)\end{bmatrix} = 1.$$

Note that for simplicity, we do not assume an explicit setup algorithm. In the following, all global parameters depend on 1^κ. We stress that we do not rely on a trusted setup authority

2.13.2 Ring Unforgeability

When describing the ring signature scheme in Chapter 6, we concentrate on unforgeability against chosen subring attacks. This security notion is formalized in the following attack game between a challenger and an adversary.

Setup. The challenger runs $\mathsf{RSIG.KeyGen}$ n times to compute $(SK_1, PK_1), \ldots, (SK_n, PK_n)$. Next, $R = (PK_1, PK_2, \ldots, PK_n)$ is given to the adversary.

Adaptive signature queries. The adversary adaptively sends q signature queries to the challenger. For $i \in \{1, \ldots, q\}$, each query Q_i consists of a message m_i, a set $R_i \subseteq R$ of public keys and an index $e_i \in \{1, \ldots, n\}$. When the challenger receives the i'th query $Q_i = (m_i, R_i, e_i)$, he computes $\sigma_i = \mathsf{RSIG.Sign}(R_i, SK_{e_i}, m_i)$ and sends it to the adversary.

Output. The attacker outputs (m^*, R^*, σ^*) with $m^* \notin \{m_1, \ldots, m_q\}$.[2]

We denote the success probability of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win the above game as $Adv_{\mathsf{SIG},\mathcal{A},\mathrm{unf}}$.

Definition 2.33 (Ring unforgeability) *We say that a ring signature scheme is (t, ϵ, q)-secure, if no t-time attacker has success probability at least ϵ in the above attack game after making q signature queries.*

2.13.3 Ring Anonymity

The strongest notion of anonymity for ring signature schemes is perfect anonymity. Formally, we consider the following attack game between a challenger and an *unbounded* adversary.

Setup. The challenger runs $\mathsf{RSIG.KeyGen}$ n times to compute $(SK_1, PK_1), \ldots, (SK_n, PK_n)$. The set of the so computed public keys $R = (PK_1, PK_2, \ldots, PK_n)$ is given to the adversary.

Adaptive signature and corrupt queries. The adversary adaptively sends q signature queries $Q_1, \ldots Q_q$ to the challenger and receives the corresponding answers $\sigma_1, \ldots, \sigma_q$.

[2] We note that a ring signature scheme which is secure under this security definition can easily be adapted to meet the slightly stronger security notion in [14] which solely requires $(m^*, R^*) \notin \{(m_1, R_1), \ldots, (m_q, R_q)\}$: given message m and subring R we simply sign $\bar{m} = h(h(m)||h(R))$ instead of m where $h(\cdot) := \mathsf{HF.Eval}(k_{\mathsf{HF}}, \cdot)$ is a collision-resistant hash function with $k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}(1^\kappa)$. For any new (m^*, R^*), $\bar{m}^*$ will now be distinct from all previous values.

At the same time, the adversary may adaptively query up to n secret keys SK_i with $i \in \{1, \ldots, n\}$.

Output. Finally, the attacker outputs a message m^*, a set of public keys $R^* \subseteq R$ and two distinct indices $i_0, i_1 \in \{1, \ldots, n\}$ such that $PK_{i_0}, PK_{i_1} \in R^*$. The challenger randomly chooses $b \in \{0, 1\}$, computes $\sigma^* = \mathsf{RSIG.Sign}(m^*, R^*, SK_{i_b})$, and sends σ^* to the attacker. The attacker then outputs b', indicating his guess for b.

We denote the advantage of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win the above game as

$$Adv_{\mathsf{RSIG},\mathcal{A},\mathrm{ano}} = \left|\Pr[\mathcal{A} \text{ outputs } b' = b] - \Pr[\mathcal{A} \text{ outputs } b' \neq b]\right| .$$

Definition 2.34 (Perfect ring anonymity) *We call a ring signature scheme perfectly anonymous, if even an unbounded adversary has no advantage ($Adv_{\mathsf{RSIG},\mathcal{A},ano} = 0$) in winning the above game.*

Chapter 3

Twin Signatures, Revisited

In 2001, Naccache, Stern and Pointcheval presented a signature scheme that is proven secure without random oracles in the standard model. The scheme is secure solely under the Strong RSA assumption. The basic idea is to securely combine two distinct RSA groups such that an attacker must break the SRSA assumption in at least one of them. However, the original scheme was defined in SRSA groups where each group element has a 1024 bit representation. It is therefore not competitive with state-of-the-art signature schemes which have signature sizes of 340 bits [17] or below [64]. We generalize the Naccache *et al.* scheme and improve its efficiency in several ways.

CONTRIBUTION. First, we present a new twin signature scheme that is based on the SDH assumption in bilinear groups. Since the representation of group elements can be much shorter in bilinear groups (≈ 170 bits) than in RSA groups (≈ 1024 bits), the new signature scheme allows for much shorter signatures.

Second, we analyze the efficiency of twin signature schemes while concentrating on the two main issues signature size and computational complexity of the signing and verification procedure. We reveal a severe efficiency bottleneck in the Naccache *et al.* scheme when signing long messages as in the presented form, the signature size is proportional to the length of the input message. Finally, we give a solution to this problem by designing a new provably secure chaining function.

3.1 Twin Signatures

One of the theoretically most important signature transformations was presented in 1996 by Cramer *et al.* [44]. It maps EMUF-RMA-secure signature schemes to EMUF-aCMA-secure schemes. In contrast to the transformation by Even, Goldreich, Micali [51], the Cramer *et al.* construction is very efficient. Given an EMUF-RMAsecure signature scheme it outputs a fully secure scheme which is only slightly less efficient. Unfortunately to the best of our knowlegde there is no natural example of a signature scheme which is secure against random message attacks but not at the same time also secure against generic chosen message attacks. In 2001 Naccache, Pointcheval, and Stern (NPS) [80] presented an efficient instantiation of the signature transformation by Cramer *et al.* using an SRSA-based weakly secure signature scheme.[1] Seemingly because of its inherent symmetry they coined the term 'twin signature' to describe their scheme. The scheme is very efficient although it cannot compete with the schemes by Gennaro *et al.* [56] and Fischlin [54].[2]

In this chapter we provide an SDH-based variant of the twin signature scheme which yields much smaller signatures than the NPS scheme. Next we present a careful analysis of how to correctly use collision-resistant hash functions for domain extension in twin signature schemes. When signing long messages (i.e. > 160 Bits), the signature size ($2l_m + 2048$) of the original scheme depends on the message size l_m. Our main result is a slight modification of the NPS scheme that reduces the signature size to $2k + 2048$ where k is the output length of the hash function ($k \approx 160$). If the hash function outputs randomly distributed output values for randomly distributed input values we can even further compress the signature size to $k + 2048$.

In the first step, we present a generic variant $\mathcal{S}_{\mathrm{SRSA}}$ of the SRSA-based twin signature scheme that was proposed by Naccache, Pointcheval, and Stern (NPS) [80] using chaining functions as introduced in Section 2.5. We will later show how to improve the original NPS scheme by introducing a new, specially crafted chaining function.

3.1.1 Generic SRSA-Based Twin Signature Scheme

Let $\mathcal{R} = \{0,1\}^{l_r}$, $\mathcal{M} = \{0,1\}^{l_m}$, and $\mathcal{W} = \{\mathcal{W}_\kappa\}_{\kappa\in\mathbb{N}}$ be a $(t_{\mathrm{chain}}, \epsilon_{\mathrm{chain}})$-chaining function. Additionally, let $\mathsf{toPrime} : \{0,1\}^{l_r} \rightarrow \mathbb{N}$ be an efficient function that injectively maps strings to prime numbers (see Section 2.2.2).

SIG.KeyGen(1^κ)**.** Choose two RSA moduli $n_1 = p_1q_1$ and $n_2 = p_2q_2$. Then randomly choose $u_1 \in \mathbb{Z}^*_{n_1}$ and $u_2 \in \mathbb{Z}^*_{n_2}$. Draw $w \overset{\$}{\leftarrow} \mathcal{W}_\kappa$. The public key is $PK = (n_1, n_2, u_1, u_2, w)$ and the secret key consists of the factorization of n_1 and n_2: $SK = (p_1, q_1, p_2, q_2)$. We require $2^{l_r} - 1 \leq \min\{\phi(n_1), \phi(n_2)\}$.

SIG.Sign(SK, m)**.** Given a secret key SK and a message $m \in \mathcal{M}$, pick $r \overset{\$}{\leftarrow} \mathcal{R}$ and compute the signature $\sigma = (s_1, s_2, r)^3$ as

$$s_1 = u_1^{\frac{1}{e_1}} \bmod n_1,\ \ s_2 = u_2^{\frac{1}{e_2}} \bmod n_2,$$

where

$$e_1 = \mathsf{toPrime}\left(w(r,m)\right) \text{ and } e_2 = \mathsf{toPrime}\left(r\right).$$

[1]Note that by definiton security under generic chosen message attacks implies security under random message attacks.

[2]In the case of [56], this might be due to the fact that the scheme implicitly use the Shamir-Tauman signature transformation [96] to map the weakly secure signature scheme to a fully secure schemes. This transformation seems to nicely exploit the additional security guarantees provided by EMUF-gCMA-secure schemes in contrast to EMUF-RMA-secure schemes.

[3]We made a small simplification here. In the original paper, not r is part of the signature but the randomness ω that is used to generate r. We stress that since r is drawn uniformly at random from $\mathcal{R}$ we must have that $|\omega| \geq l_r = 2l_m$ and therefore ω is also dependent on the message size.

SIG.Verify(PK, m, σ)**.** Given a public key PK, a message m, and a signature $\sigma = (s_1, s_2, r)$, verify if the following equations hold:

$$s_1^{e_1} \overset{?}{=} u_1 \bmod n_1,\ s_2^{e_2} \overset{?}{=} u_2 \bmod n_2,$$

where

$$e_1 = \mathsf{toPrime}\left(w(r,m)\right) \text{ and } e_2 = \mathsf{toPrime}\left(r\right).$$

Theorem 3.1 *Suppose the* $(t_{SRSA}, \epsilon_{SRSA})$*-SRSA assumption holds and* $\mathcal{W} = \{\mathcal{W}_\kappa\}_{\kappa\in\mathbb{N}}$ *is a* $(t_{chain}, \epsilon_{chain})$*-chaining function. Then,* $\mathcal{S}_{SRSA}$ *is* (q, t, ϵ)*-secure against adaptive chosen message attacks provided that*

$$t \approx t_{SRSA} + t_{chain},\ \epsilon \leq 2\epsilon_{SRSA} + 2q^2\epsilon_{chain}.$$

The proof of Theorem 3.1 closely follows [80].

3.2 An SDH Based Twin Signature Scheme

In this section, we present our SDH based twin signature scheme $\mathcal{S}_{\text{SDH}}$. Let a bilinear group $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ and a $(t_{\text{chain}}, \epsilon_{\text{chain}})$-chaining function $\mathcal{W} = \{\mathcal{W}_\kappa\}_{\kappa\in\mathbb{N}}$ with $\mathcal{R} = \{0, \ldots, t_r\}$ and $t_r \leq p$ be given. Let $g_T = e(g_1, g_2)$.

SIG.KeyGen(1^κ)**.** Choose two elements $x_1, x_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$. Draw $w \overset{\$}{\leftarrow} \mathcal{W}_\kappa$. The public key is $PK = (PK_1, PK_2, w)$ with $PK_1 = g_2^{x_1}$ and $PK_2 = g_2^{x_2}$. The secret key is $SK = (x_1, x_2)$.

SIG.Sign(SK, m)**.** Given a secret key SK and a message $m \in \mathcal{M}$, pick $r \overset{\$}{\leftarrow} \mathcal{R}$ with $r \neq -x_2$ and $w(r,m) \neq -x_1$ and compute $\sigma = (s_1, s_2, r)$ as

$$s_1 = g_1^{\frac{1}{(x_1 + w(r,m))}},\ s_2 = g_1^{\frac{1}{(x_2+r)}}.$$

SIG.Verify(PK, m, σ)**.** Given a public key PK a message m and a signature $\sigma = (s_1, s_2, r)$, verify whether the following equations hold:

$$e\left(s_1, PK_1 \cdot g_2^{w(r,m)}\right) \overset{?}{=} g_T,\ e\left(s_2, PK_2 \cdot g_2^{r}\right) \overset{?}{=} g_T.$$

3.3 Security

Theorem 3.2 *Suppose that the* $(q_{SDH}, t_{SDH}, \epsilon_{SDH})$*-SDH assumption holds and that* $\mathcal{W}$ *is a* $(t_{chain}, \epsilon_{chain})$*-chaining function. Then,* $\mathcal{S}_{SDH}$ *is* (q, t, ϵ)*-secure against adaptive chosen message attacks with*

$$q = q_{SDH},\ t \approx t_{SDH} + t_{chain},\ \epsilon \leq 2\epsilon_{SDH} + 2q^2\epsilon_{chain}.$$

Proof. By contradiction. Assume algorithm $\mathcal{A}$ is a forger that (q,t,ϵ)-breaks the existential unforgeability of $\mathcal{S}_{\text{SDH}}$. Then, we can construct a simulator $\mathcal{B}$, that interacts with $\mathcal{A}$ and after q signature queries either breaks the SDH problem or the security of the chaining function in time t with advantage ϵ. We consider four types of forgeries $m^*, (s_1^*, s_2^*, r^*)$ $\mathcal{A}$ can output after making q signature queries $m_1, \ldots, m_q$ and receiving q responses $(s_{1,1}, s_{2,1}, r_1), \ldots, (s_{1,q}, s_{2,q}, r_q)$:
Type I: $r^* \notin \{r_1, \ldots, r_q\}$.
Type II: $r^* = r_i$, $i \in \{1, \ldots, q\}$, $w(r^*, m^*) \notin \{w(r_1, m_1), \ldots, w(r_q, m_q)\}$.
Type III: $r^* = r_i$, $w(r^*, m^*) = w(r_j, m_j)$, $i, j \in \{1, \ldots, q\}$, $i \neq j$.
Type IV: $r^* = r_i$, $w(r^*, m^*) = w(r_i, m_i)$, $i \in \{1, \ldots, q\}$.

We subsequently assume that $\mathcal{B}$ at the beginning of the communication guesses which type of forgery $\mathcal{A}$ is going to output (with probability $\frac{1}{4}$). According to this guess, $\mathcal{B}$ proceeds differently. If $\mathcal{B}$'s guess turns out to be wrong, he simply aborts and restarts. Theorem 3.2 then follows by a standard hybrid argument.

Let $\left(\hat{g}_1, \hat{g}_1^x, \hat{g}_1^{(x^2)}, \ldots, \hat{g}_1^{(x^q)}, \hat{g}_2, \hat{g}_2^x\right)$ be the SDH challenge and $(\hat{m}, \hat{r}, \hat{r}')$ the challenge for Property 4 of the chaining function. In the following, we provide a proof of security that proceeds in a sequence of games [98, 11]. Let $\Pr[S_i]$ denote the success probability for an attacker to successfully forge signatures in Game i.

Type I Forger
Game$_0$. This is the original attack game. By assumption, $\mathcal{A}$ (q,t,ϵ)-breaks $\mathcal{S}$ when interacting with the signing oracle $\mathcal{O}_{SK}(\cdot)$. We have

$$\Pr[S_0] = \epsilon \ . \tag{3.1}$$

Game$_1$. This game is like the previous one except that $\mathcal{B}$ *constructs* the values g_1, PK_1 and PK_2 using the SDH challenge. First, $\mathcal{B}$ chooses $x_1 \xleftarrow{\$} \mathbb{Z}_p$ and $r_1, \ldots, r_q \xleftarrow{\$} \mathcal{R}$. Then, it computes $g_1 = \hat{g}_1^{r(x)}$, $g_2 = \hat{g}_2$, $PK_1 = g_2^{x_1}$, $PK_2 = g_2^x$, where $r(x) := \prod_{i=1}^q (x + r_i)$. Note that $\mathcal{B}$ can easily generate g_1 by computing the coefficients $\beta_i \in \mathbb{Z}$ for $i \in \{0, \ldots, q\}$ of $r(x) = \sum_0^q \beta_i x^i$ and evaluating $\hat{g}_1^{r(x)} = \prod_{i=0}^q \left(\hat{g}_1^{(x^i)}\right)^{\beta_i}$. Since x_1 and the r_i's are chosen at random, the distribution of the constructed values is equal to the previous game. Thus,

$$\Pr[S_1] = \Pr[S_0] \ . \tag{3.2}$$

Game$_2$. Now, $\mathcal{B}$ simulates $\mathcal{O}_{SK}(\cdot)$ by answering $\mathcal{A}$'s signature queries. For each received message $m_j \in \mathcal{M}$, $\mathcal{B}$ outputs $\sigma_j = (s_{1,j}, s_{2,j}, r_j)$ with

$$s_{1,j} = g_1^{1/(x_1 + w(r_j, m_j))}, \ s_{2,j} = \hat{g}_1^{\prod_{i=1, i\neq j}^q (x + r_i)} .$$

The simulator can easily compute $s_{1,j}$ since it knows x_1. The value $s_{2,j}$ can be computed like g_1 using the SDH challenge. Since the r_j's have been chosen randomly, the changes are purely conceptual.

$$\Pr[S_2] = \Pr[S_1] \ . \tag{3.3}$$

Let us now consider the forgery (m^*, s_1^*, s_2^*, r^*) output by $\mathcal{A}$. It must hold that

$$e(s_1^*, PK_1 g_2^{w(r^*,m^*)}) = e(s_2^*, PK_2 g_2^{r^*}) = e(g_1, g_2) \ .$$

By assumption, we have that $r^* \notin \{r_1, \ldots, r_q\}$. Using long division, the simulator can now compute $d \in \mathbb{Z}$ and a polynomial $r'(x)$ of degree $q-1$ with coefficients in $\mathbb{Z}$ such that $r(x) = (x + r^*)r'(x) + d$. Since $(x + r^*)$ does not divide $r(x)$ we must have that $d \neq 0$. From the second verification equation we get that $s_2^* = g_1^{1/(x+r^*)} = \hat{g}_1^{r'(x)+d/(x+r^*)}$. Therefore, $\mathcal{B}$ can compute a solution to the SDH problem as

$$\left(\left(s_2^* \hat{g}_1^{-r'(x)}\right)^{1/d} = \hat{g}_1^{1/(x+r^*)}, r^*\right) .$$

We finally have that

$$\Pr[S_2] = \epsilon_{\text{SDH}} \ . \tag{3.4}$$

Type II Forger
Now suppose $\mathcal{B}$ correctly expects $\mathcal{A}$ to be a Type II Forger. The proof proceeds analogously to the proof for Type I Forgers.

Game$_1$. In this game, $\mathcal{B}$ chooses random $x_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $w_1, \ldots, w_q \overset{\$}{\leftarrow} \mathcal{R}$. Now define $v(x) := \prod_{i=0}^{q}(x + w_i)$. Then, $\mathcal{B}$ computes

$$g_1 = \hat{g}_1^{v(x)},\ g_2 = \hat{g}_2,\ PK_1 = g_2^x,\ PK_2 = g_2^{x_2}.$$

As before, the distribution of the values is equal to the previous game.

$$\Pr[S_1] = \Pr[S_0] \ . \tag{3.5}$$

Game$_2$. In this game, $\mathcal{B}$ simulates the signing oracle. When receiving a query $m_j \in \mathcal{M}$, it first computes $r_j \in \mathcal{R}$ such that $w_j = w(r_j, m_j)$. It then outputs the signature $\sigma_j = (s_{1,j}, s_{2,j}, r_j)$ with

$$s_{1,j} = \hat{g}_1^{\prod_{i=1, i\neq j}^{q}(x+w_i)},\ s_{2,j} = g_1^{1/(x_2+r_j)}.$$

Since the w_i's are chosen uniformly at random from $\mathcal{R}$, so are the r_i's because w is a permutation in the first input parameter.

$$\Pr[S_2] = \Pr[S_1] \ . \tag{3.6}$$

Similar to the previous case, $\mathcal{B}$ can compute a polynomial $v'(x)$ and $d \in \mathbb{Z}$ with $d \neq 0$ such that $v(x) = (x + w(r^*, m^*))v'(x) + d$ because $w(r^*, m^*) \notin \{w(r_1, m_1), \ldots, w(r_q, m_q)\}$ and thus $(x + w(r^*, m^*))$ does not divide $v(x)$. The simulator then outputs a solution to the SDH problem as

$$\left(\left(s_2^* \hat{g}_1^{-v'(x)}\right)^{1/d}, w(r^*, m^*)\right) .$$

We have that

$$\Pr[S_2] = \epsilon_{\text{SDH}} \ . \tag{3.7}$$

Type III Forger

Game$_1$. In this game, we let the simulator choose both secret keys $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$. It sets $g_1 = \hat{g}_1$, $g_2 = \hat{g}_2$, $PK_1 = g_2^{x_1}$, and $PK_2 = g_2^{x_2}$.

$$\Pr[S_1] = \Pr[S_0] \ . \tag{3.8}$$

Game$_2$. Next, $\mathcal{B}$ makes two random guesses for distinct indices $j, k \xleftarrow{\$} \{1, \ldots, q\}$. For all $i \in \{1, \ldots, q\} \setminus \{j, k\}$, the simulator proceeds according to SIG.Sign to simulate $\mathcal{O}_{SK}(\cdot)$ using the secret keys x_1 and x_2. In particular, this means that the r_i are chosen uniformly at random from $\mathcal{R}$. For the j'th signature, $\mathcal{B}$ *computes* r_j such that $w(\hat{r}, \hat{m}) = w(r_j, m_j)$ using the challenge for the chaining function. For the k'th signature, $\mathcal{B}$ sets $r_k = \hat{r}'$. Since $\hat{r}$ and $\hat{r}'$ originate from a random challenge for the chaining function and the remaining r_i are chosen randomly we still have that

$$\Pr[S_2] = \Pr[S_1] \ . \tag{3.9}$$

Game$_3$. By assumption, we have that $\mathcal{A}$ outputs a forgery $m^*, (s_1^*, s_2^*, r^*)$ with $r^* = r_a$ for some $a \in \{1, \ldots, q\}$ and $w(r^*, m^*) = w(r_b, m_b)$ for some $b \in \{1, \ldots, q\}$. We have that $a \neq b$. Now, $\mathcal{B}$ aborts if $b \neq j$ or $a \neq k$. Since r_j and r_k are perfectly indistinguishable from the remaining r_i we have that

$$\Pr[S_3] \ = \ 1/q^2 \cdot \Pr[S_2] \ . \tag{3.10}$$

Otherwise, $\mathcal{B}$ has found a solution to the challenge for Property 4

$$w(\hat{r}', m^*) = w(r_k, m^*) = w(r_a, m^*) = w(r^*, m^*) = w(r_b, m_b) = w(\hat{r}, \hat{m}).$$

$$\Pr[S_3] \ = \ \epsilon_{\text{chain}} \ . \tag{3.11}$$

Type IV Forger

Game$_1$. As before, we let the simulator choose both secret keys $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$. It sets $g_1 = \hat{g}_1$, $g_2 = \hat{g}_2$, $PK_1 = g_2^{x_1}$, and $PK_2 = g_2^{x_2}$. Clearly,

$$\Pr[S_1] = \Pr[S_0] \ . \tag{3.12}$$

Game$_2$. To simulate the signing oracle, $\mathcal{B}$ simply uses x_1 and x_2 as specified by SIG.Sign. Hence,

$$\Pr[S_2] = \Pr[S_1] \ . \tag{3.13}$$

By assumption we have that $\mathcal{B}$ $w(r^*, m^*) = w(r_j, m^*) = w(r_j, m_j)$ for some $j \in \{1, \ldots, q\}$. From the definition of the attack game we also know that $m^* \neq m_j$. So, $\mathcal{B}$ has found a collision for the chaining function (Property 3) and

$$\Pr[S_2] = \epsilon_{\text{chain}} \ . \tag{3.14}$$

Putting together Equations (3.1)-(3.14), we get that $\epsilon = \epsilon_{\text{SDH}}$ for Type I and Type II forgeries, $\epsilon = q^2\epsilon_{\text{chain}}$ for Type III forgers, and $\epsilon = \epsilon_{\text{chain}}$ for Type IV forgers. Clearly, $\epsilon \leq \epsilon_{\text{SDH}} + q^2\epsilon_{\text{chain}}$ for each type of forger. Finally, considering $\mathcal{B}$'s initial guess we have that $\epsilon \leq 2\epsilon_{\text{SDH}} + 2q^2\epsilon_{\text{chain}}$ what proofs Theorem 3.2. □

3.4 Improved Chaining Function

In this section, we analyze the efficiency of twin signature schemes when signing long messages. We concentrate on two issues. The first issue concerns the computational efficiency of the scheme. Recall the chaining function from Section 2.5 (Lemma 2.20). Since in the SRSA based scheme the input of p and hence the computational efficiency depends on the message size l_m, Naccache *et al.* recommend to apply a hash function $h(\cdot) := \mathsf{HF.Eval}(k_{\mathsf{HF}}, \cdot)$ with $k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}$ prior to the prime mapping function $\mathsf{toPrime}$ [80, §B.2]: $w'(r,m) = h(r \oplus (m||m))$. Now, $\mathsf{toPrime}$ must only map values of maximal length k to prime numbers. One can easily see, that this improvement does not address the signature size, since r, by condition $l_r = 2l_m$, still depends on l_m. This is our second issue. A further improvement is to first apply the collision-resistant hash function *to the input message* and then to sign the resulting hash value: $w''(r,m) = r \oplus (h(m)||h(m))$. Now, we have that $l_r = 2k$ and, as a result, r and the signature size are independent of l_m. For both improvements (w' and w'') we must assume that the underlying hash function is collision-resistant and so rely on additional complexity assumptions to maintain provable security (as long as the hash function does not solely rely on complexity assumptions that are provably weaker than SRSA, SDH respectively). We stress that we use hash functions solely for the purpose of (message) domain extension, not as instantiations of random oracles. For practical parameters, say $k = 160$, we now have that $l_r = 320$. The final question is if this situation can further be improved. Ideally, we would have that $l_r = 160$. This has several benefits. First, prime mapping can be twice as fast. Second, the signature size could be reduced by 160 bits. Surprisingly, we can show that when using hash functions with the additional property that random input strings are mapped to uniformly distributed output strings, we can do without the string concatenation:

$$\tilde{w} : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$$
$$\tilde{w}(r,m) = r \oplus h(m) \ .$$

Lemma 3.3 *Let* $\mathsf{HF} = (\mathsf{HF.KeyGen}, \mathsf{HF.Eval})$ *be a* $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$*-collision-resistant hash function. Furthermore, assume that for every* $k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}(1^\kappa)$ $\mathsf{HF.Eval}(k_{\mathsf{HF}}, m)$ *is uniformly distributed in* $\{0,1\}^k$ $(k = k(\kappa))$ *for random* $m \in \{0,1\}^*$. *Then,* $\mathcal{W} = \{\mathcal{W}_k\}$ *with* $\mathcal{W}_k = \{\tilde{w}(r,m) = r \oplus \mathsf{HF.Eval}(k_{\mathsf{HF}}, m) |\, k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}(1^\kappa)\}$ *is* $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$*-chaining.*

PROOF. The first two properties can be verified by inspection. Let again $h(\cdot) := \mathsf{HF.Eval}(k_{\mathsf{HF}}, \cdot)$. Property 3 holds because $\tilde{r} \oplus h(m) = \tilde{r} \oplus h(m')$ implies $h(m) = h(m')$. So, any attacker that breaks Property 3 also breaks the collision-resistance of the hash function. To show that $\tilde{w}$

Table 3.1: Efficiency Comparison of Chaining Functions. Elements in $\mathbb{G}_1$ have around 170 bits [18, 79], while RSA moduli have at least 1024 bits. Let $l_{\max} = \max\{4l_m, 2048\}$.

scheme	chaining function	signature size in bits	#prim. tests	#pairings
$\mathcal{S}_{\text{SDH}}$	$r \oplus h(m)$	$k + 340$	–	2
$\mathcal{S}_{\text{SRSA}}$	$r \oplus h(m)$	$k + 2048$	$O(k)$	–
$\mathcal{S}_{\text{SRSA}}$	$r \oplus (h(m)\|\|h(m))$	$2k + 2048$	$O(k)$	–
[80, §B.2]	$h(r \oplus (m\|\|m))$	$2l_m + 2048$	$O(k)$	–
[80]	$r \oplus (m\|\|m)$	$2l_m + l_{\max}$	$O(l_m)$	–

fulfills Property 4 observe that $\tilde{r} \oplus h(m) = \tilde{r}' \oplus h(m')$ implies $\tilde{r}' \oplus \tilde{r} \oplus h(m) = h(m')$ for given random $r, r' \in \{0,1\}^k$ and $m \in \{0,1\}^*$. Now, for contradiction, assume there exists an attacker $\mathcal{A}$ that can find m' with probability $\geq \epsilon_h$ in time t_h. Then we can construct algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to find collisions in the hash function. At first $\mathcal{B}$ chooses uniformly at random $\bar{m} \in \{0,1\}^*$ and computes $y = h(\bar{m})$. Next, $\mathcal{B}$ chooses a random $m \in \{0,1\}^*$ and a random $r \in \{0,1\}^k$, computes $h(m)$, $r' = y \oplus h(m) \oplus r$ and runs $\mathcal{A}$ on (m, r, r'). As $h(m)$ is random in $\{0,1\}^k$, so is r'. Finally, $\mathcal{A}$ outputs m' and since we have that $r' \oplus r \oplus h(m) = h(m') = y = h(\bar{m})$ and because $\bar{m} \neq m'$ with overwhelming probability, $\mathcal{B}$ has found a hash collision. This contradicts the fact that HF is $(t_{\mathsf{HF}}, \epsilon_{\mathsf{HF}})$-collision-resistant. □

3.5 Offline/Online Signature Schemes

In this section, we present a modification that further improves the efficiency of $\mathcal{S}_{\text{SRSA}}$ and $\mathcal{S}_{\text{SDH}}$ by turning them into offline/online signature schemes [51]. The key idea is to precompute s_1 using a random $z \in \mathcal{R}$ in the offline phase as $s_1 = u_1^{1/z} \bmod n_1$, $s_1 = g_1^{1/(x_1+z)}$ respectively. In the online phase, when the signer wants to sign message m, she only has to find r such that $z = w(r, m)$ and compute $s_2 = u_2^{1/(r)} \bmod n_2$, $s_2 = g_1^{1/(x_2+r)}$ respectively. In this way, we can save costly inversion and exponentiation operations in the online phase. However, the efficiency gain seems rather small when compared with what can be achieved in offline/online signature schemes that are based on signature schemes which utilize chameleon hash functions [70], like for example [96].

Chapter 4

Tight Proofs for Signature Schemes without Random Oracles

It is no secret why cryptographers are interested in tight security proofs: besides being theoretically interesting, they allow for shorter security parameters and better efficiency. The research which led to the results of this chapter was also motivated by the observation that for several of the existing Strong RSA based signature schemes without random oracles we do not know if tight security proofs exist. Those schemes which we know to have a tight security proof, also have some limitations concerning practicability (which in turn cannot be found among the signature schemes with a loose security reduction). In 2007, Chevallier-Mames and Joye addressed this problem in the following way [39]: they took a tightly secure signature scheme, the Gennaro-Halevi-Rabin scheme [56], and improved its efficiency by re-designing one of its most time-consuming functions.[1] The problem with such an approach is that it only affects *new* implementations of the considered signature scheme. Therefore, we take the same approach as Bernstein at EUROCRYPT '08 who proved tight security for the *original* Rabin-Williams signature scheme in the random-oracle model [15]. However, in contrast to Bernstein we concentrate on schemes that are secure in the standard model.

Contribution. This chapter deals with the following question: are there tight security proofs for the existing practical signature schemes by Cramer-Shoup [45], Zhu [104], Camenisch-Lysyanskaya [30] and Fischlin [54] (which we only know to have loose security reductions)? We answer this question in the affirmative and present the first tight proofs for the above signature schemes. However, our result is not limited to the original schemes. In our analysis, we generalize the schemes by Camenisch-Lysyanskaya, Fischlin and Zhu by introducing a new family of randomization functions, called combining functions. The result of this generalization is an abstract signature scheme termed 'combining scheme'. In a similar way, we introduce a second general class of signature schemes called 'chameleon hash scheme' that can be regarded as a generalization of the Cramer-Shoup signature scheme.

[1]Basically, they introduced a new method to map messages to primes which is much more efficient in the verification process than the original method from [56] by *choosing* a random prime and making use of a chameleon hash function to map the input message to that prime.

Then, we prove the combining signature scheme and the chameleon hash scheme to be *tightly* secure under the SRSA assumption when instantiated with any secure combining function, respectively chameleon hash function.[2] Finally, we show that our results do not only hold under the SRSA assumption. We analyze whether there also exist tight security reductions for analogous schemes based on the SDH assumption in bilinear groups. Interestingly, most of the above schemes have not been considered yet under the SDH assumption (except for the Camenisch-Lysyanskaya scheme), although, at the same security level, the group description is much shorter in bilinear groups than in factoring based groups. We develop a SDH based variant of the combining signature scheme and the chameleon hash scheme and prove it to be existentially unforgeable under adaptive chosen message attacks with a *tight* security reduction. In doing so, we present the first SDH based variants of the Fischlin, the Zhu and the Cramer-Shoup signature scheme and the first tight security proof of the SDH based Camenisch-Lysyanskaya scheme. When instantiated with existing combining functions (respectively chameleon hash functions), we obtain short and efficient signature schemes. Our results can be interpreted in two positive ways: 1) Existing implementations of the affected signature schemes (with a fixed parameter size) provide higher security than expected. 2) New implementations can have shorter security parameters what transfers to higher efficiency.

Technical Contribution. In the existing proofs, the simulator partitions the set of forgeries by at first guessing $j \in \{1, \ldots, q\}$ where q is the number of signature queries made by the attacker. Only if the attacker's forgery shares some common values with the answer to the j-th signature query the simulator can break the SRSA assumption. Otherwise the simulator just aborts. The number of signature queries rises polynomially in the security parameter and the security proof loses a factor of q here. Our main contribution is a new technique that renders the initial guess unnecessary. As a consequence, *any* forgery helps the simulator to break the SRSA assumption. This results in a tight security proof.

Related Work. Our work is related to the existing hash-and-sign signature schemes without random oracles that are proven secure under the SRSA or the SDH assumption. Table 4.1 gives an overview on the available results.

4.1 Settings

For convenience, we also describe two general setup and key generation procedures (settings) in Section 4.1.1, and Section 4.1.2. When describing our signature schemes in Sections 4.2.1, 4.2.2, 4.2.5 we will refer to the corresponding setting and only describe the signature generation and verification algorithms.

[2]Unfortunately the security proof of the SRSA based chameleon hash scheme does not directly transfer to the Cramer-Shoup signature scheme. This is simply because in the Cramer-Shoup scheme the keys of the chameleon hash function are not chosen independently. Nevertheless, the proof of the Cramer-Shoup signature scheme is technically very similar to the proof of the chameleon hash scheme. For completeness, we also provide a full proof of (tight) security of the Cramer-Shoup signature scheme in Section 4.3.5.

Signature Scheme	Security Assumption	Security Loss		Prime Generation
		Original Reduction	Our Reduction	
Gennaro-Halevi-Rabin [56]	SRSA	O(1)		**INJ**
Cramer-Shoup [45]	SRSA	O(q)	O(1)	
Naccache-Pointcheval-Stern [80]	SRSA	O(1)		**INJ**
Fischlin [54]	SRSA	O(q)	O(1)	
Zhu [104]	SRSA	O(q)	O(1)	
Camenisch-Lysyanskaya [30]	SRSA	O(q)	O(1)	
Chevallier-Mames-Joye [39]	SRSA	O(1)		
Hofheinz-Kiltz [64]	SRSA	O(q)		
Boneh-Boyen [17]	SDH	O(1)		
Camenisch-Lysyanskaya [31, 85, 2]	SDH	O(q)	O(1)	
Hofheinz-Kiltz [64]	SDH	O(q)		

Table 4.1: Tightness of SRSA and SDH based signature schemes. **INJ** indicate that the signature scheme requires an injective mapping of messages to primes. As a consequence the verifier must perform $O(|m|_2)$ primality test to find a prime.

4.1.1 The Strong RSA Setting

Definition 4.1 (SRSA setting) *In this setting,* $\mathsf{SIG.KeyGen}(1^\kappa)$ *outputs* $(SK = (p, q),$ $PK = n)$ *for a safe modulus* $n = pq$ *such that* $p = 2p' + 1$, $q = 2q' + 1$, *and* p, q, p', q' *are primes. All computations are performed in the cyclic group* QR_n. *Let* $l_i = l_i(\kappa)$ *for* $i \in \{n, t, c, e, m\}$ *be polynomials. We require that* $|n|_2 = l_n$ *and* $|p'|_2 = |q'|_2 = l_n/2 - 1$. *Furthermore, we assume that the* $(t_{SRSA}, \epsilon_{SRSA})$*-SRSA assumption holds. We let* u, v, w *be public random generators of* QR_n *with unknown* $\log_u v$, $\log_u w$, *and* $\log_v w$. *When using combining functions* $z(r, m)$, *we assume that* $\mathcal{M} \subseteq [0; 2^{l_m} - 1]$, $\mathcal{Z} \subseteq [0; 2^{l_z} - 1]$ *and* $\mathcal{R} \subseteq [0; 2^{l_r} - 1]$. *We let* $E \subseteq [2^{l_e - 1}; 2^{l_e} - 1]$ *denote the set of* l_e*-bit primes. Finally, we require that* $l_m \leq l_c, l_z, l_r < l_e < l_n/2 - 1$.

4.1.2 The Strong Diffie-Hellman Setting

Definition 4.2 (SDH setting) *Let* $l_p = l_p(\kappa)$ *be a polynomial. In the SDH setting, all computations are performed in the cyclic groups of* $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ *with* $|p|_2 = l_p$. *The PPT* $\mathsf{SIG.KeyGen}(1^\kappa)$ *chooses* $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ *and outputs* $(SK = x, PK = g_2^x)$. *We assume that the* $(q_{SDH}, t_{SDH}, \epsilon_{SDH})$*-SDH assumption holds. Finally, we suppose that the values* $a, b, c \in \mathbb{G}_1$ *are public random generators of* $\mathbb{G}_1$ *such that* $\log_a b$, $\log_a c$, *and* $\log_b c$ *are unknown. combining functions* $z(r, m)$, *we assume that* $\mathcal{Z} \subseteq \mathbb{Z}_p$ *and* $\mathcal{R} \subseteq \mathbb{Z}_p$.

4.2 Signature Classes

For convenience, we now introduce two general signature classes. The combining signature scheme $\mathcal{S}_{\mathrm{CMB}}$ constitutes an useful abstraction of the Camenisch-Lysyanskaya, the Fischlin, and the Zhu signature scheme using combining functions. The chameleon signature scheme $\mathcal{S}_{\mathrm{CH}}$ can be regarded as a general variant of the original Cramer-Shoup signature scheme where we do not specify a concrete instantiation of the chameleon hash function. In the following we will make use of the combining functions given in Table 2.1 of Section 2.6.

4.2.1 SRSA Based Combining Signature Scheme ($\mathcal{S}_{\mathbf{CMB,SRSA}}$)

In the SRSA setting, $\mathsf{SIG.Sign}(SK, m)$ randomly chooses $r \in \mathcal{R}$ and $e \in E$ and computes a signature $\sigma = (r, s, e)$ on message m with $s = \left(uv^r w^{z(r,m)}\right)^{\frac{1}{e}}$. Let us now show, that our construction indeed generalizes the claimed signature schemes. Observe that we can easily obtain the Fischlin scheme [54] when we instantiate the combining function with EX 2 of Table 2.1. Furthermore, we can also get the Camenisch-Lysyanskaya scheme [30] using EX 3. This becomes obvious if we substitute v by $v' = vw$ as $uv^r w^{r+m} = u(vw)^r w^m = u(v')^r w^m$. [3] We note that when we use the Camenisch-Lysyanskaya scheme with long messages we first have to apply a collision-resistant hash function to the message. What we essentially get is Zhu's scheme [103, 104]. By Lemma 2.23, the resulting function is still combining. The verification algorithm $\mathsf{SIG.Verify}(PK, m, \sigma)$ takes a purported signature $\sigma = (r, s, e)$ and checks if $s^e \overset{?}{=} uv^r w^{z(r,m)}$, if $|e|_2 = l_e$, and if e is odd.

4.2.2 SDH Based Combining Signature Scheme ($\mathcal{S}_{\mathbf{CMB,SDH}}$)

We also present a SDH based variant $\mathcal{S}_{\mathrm{CMB,SDH}}$ of the combining signature scheme. We remark that for the Camenisch-Lysyanskaya scheme there already exists a corresponding SDH based variant, originally introduced in [31] and proven secure in [85, 2]. Similar to $\mathcal{S}_{\mathrm{CMB,SRSA}}$, we obtain the SDH based Camenisch-Lysyanskaya scheme when instantiating the combining function with EX 1. In the same way, we can also get SDH based variants of the Fischlin signature scheme (using EX 2) and of Zhu's scheme (using Lemma 2.23). In the SDH based combining scheme, $\mathsf{SIG.Sign}(SK, m)$ at first chooses a random $r \in \mathcal{R}$ and a random $t \in \mathbb{Z}_p \setminus \{-x\}$. It then computes the signature $\sigma = (r, s, t)$ with $s = \left(ab^r c^{z(r,m)}\right)^{\frac{1}{x+t}}$. Given a signature $\sigma = (r, s, t)$, $\mathsf{SIG.Verify}(PK, m, \sigma)$ checks if $e\left(s, PKg_2^t\right) \overset{?}{=} e\left(ab^r c^{z(r,m)}, g_2\right)$.

[3] To be precise, our generalization slightly differs from the Camenisch-Lysyanskaya scheme. In the original scheme, it is required that $l_r = l_n + l_m + 160$. As a result, the authors recommend for 160 bit long messages that $l_r = 1346$, $l_s = 1024$, and $l_e = 162$. In our scheme, we simply require that $l_m \leq l_r < l_e < l_n/2 - 1$. Then, we can set $l_r = 320$, $l_s = 1024$, and $l_e = 321$ for a probability $\epsilon_{\mathrm{comb}} = 2^{-160}$. Therefore, the signature size of our signature scheme is much shorter (only $(320 + 1024 + 321)/(1346 + 1024 + 162) \approx 66\%$ of the original signature size) and the scheme is more efficient (since shorter exponents imply faster exponentations) than the original scheme.

Table 4.2: Comparison of signature generation and verification. We implicitly require that the verifier checks that the signature components are in the correct ranges (except for e in the SRSA setting).

		SRSA setting	SDH setting
		$e \xleftarrow{\$} E,\ r \xleftarrow{\$} \mathcal{R},\ \sigma = (r, s, e)$	$t \xleftarrow{\$} \mathbb{Z}_p \setminus \{-x\},\ r \xleftarrow{\$} \mathcal{R},\ \sigma = (r, s, t)$
Chameleon Hash	sign	$s = \left(uv^{ch(r,m)}\right)^{\frac{1}{e}}$	$s = \left(ab^{ch(r,m)}\right)^{\frac{1}{x+t}}$
	verify	$s^e \overset{?}{=} uv^{ch(r,m)},\ e$ odd?, $\|e\| = l_e$?	$e\left(s, PKg_2^t\right) \overset{?}{=} e\left(ab^{ch(r,m)}, g_2\right)$
Combining	sign	$s = \left(uv^r w^{z(r,m)}\right)^{\frac{1}{e}}$	$s = \left(ab^r c^{z(r,m)}\right)^{\frac{1}{x+t}}$
	verify	$s^e \overset{?}{=} uv^r w^{z(r,m)},\ e$ odd?, $\|e\| = l_e$?	$e\left(s, PKg_2^t\right) \overset{?}{=} e\left(ab^r c^{z(r,m)}, g_2\right)$

4.2.3 SRSA Chameleon Hash Signature Scheme ($\mathcal{S}_{\mathbf{CH,SRSA}}$)

The signature scheme $\mathcal{S}_{\mathrm{CH,SRSA}}$ is defined in the SRSA setting. $\mathsf{SIG.KeyGen}(1^\kappa)$ additionally generates the key material $(SK_{\mathsf{CH}}, PK_{\mathsf{CH}})$ for a chameleon hash function. The value PK_{CH} is added to the scheme's public key. (SK_{CH} is not required. However, it may be useful when turning the signature scheme into an online-offline signature scheme [96].) The signature generation algorithm $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}$ and a random prime $e \in E$. It then outputs the signature $\sigma = (r, s, e)$ on a message m where $s = \left(uv^{ch(r,m)}\right)^{\frac{1}{e}}$. To verify a purported signature $\sigma = (r, s, e)$ on m, $\mathsf{SIG.Verify}(PK, m, \sigma)$ checks if e is odd, if $|e|_2 = l_e$, and if $s^e \overset{?}{=} uv^{ch(r,m)}$.

4.2.4 SDH Based Chameleon Hash Signature Scheme ($\mathcal{S}_{\mathbf{CH,SDH}}$)

Let us now define a new variant of the chameleon hash signature scheme that is based on the SDH assumption. Again, $\mathsf{SIG.KeyGen}(1^\kappa)$ also adds the public key PK_{CH} of a chameleon hash function to PK. In the SDH setting, $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}$ and a random $t \in \mathbb{Z}_p \setminus \{-x\}$. Using $SK = x$, it then outputs the signature σ on m as $\sigma = (r, s, t)$ where $s = \left(ab^{ch(r,m)}\right)^{\frac{1}{x+t}}$. To verify a given signature $\sigma = (r, s, t)$ on m, $\mathsf{SIG.Verify}(PK, m, \sigma)$ checks if $e\left(s, PKg_2^t\right) \overset{?}{=} e\left(ab^{ch(r,m)}, g_2\right)$. A suitable chameleon hash function can for example be found in [70].

4.2.5 The Cramer-Shoup Signature Scheme ($\mathcal{S}_{\mathbf{CS,SRSA}}$)

Let us now review the Cramer-Shoup signature scheme that is defined in the SRSA setting. The Cramer-Shoup scheme $\mathcal{S}_{\mathrm{CS,SRSA}}$ additionally requires a collision-resistant hash function $\mathsf{HF} = (\mathsf{HF.KeyGen}, \mathsf{HF.Eval})$ with $\mathcal{M}_{\mathsf{HF}} = \{0,1\}^*$ and $\mathcal{Y}_{\mathsf{HF}} = \{0,1\}^{l_c}$. The message space is so extended to $\mathcal{M} = \{0,1\}^*$. We assume $l_c < l_e < l_n/2 - 1$.

- $\mathsf{SIG.KeyGen}(1^\kappa)$ also computes a random l_e-bit prime $\tilde{e}$ and $k_{\mathsf{HF}} \leftarrow \mathsf{HF.KeyGen}(1^\kappa)$. The secret key is $SK = (p, q)$ the public key is $PK = (n, \tilde{e}, k_{\mathsf{HF}})$.

- $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in QR_n$ and evaluates (the chameleon hash function) $c = r^{\tilde{e}}/v^{h(m)} \mod n$ where $h(\cdot) := \mathsf{HF.Eval}(k_{\mathsf{HF}}, \cdot)$. Then it draws a random l_e-bit prime $e \neq \tilde{e}$ and computes the value $s = \left(uv^{h(c)}\right)^{1/e} \mod n$. The signature is $\sigma = (r, s, e)$.

- $\mathsf{SIG.Verify}(PK, m, \sigma)$ re-computes $c = r^{\tilde{e}}/v^{h(m)} \mod n$ and checks if $s \overset{?}{=} \left(uv^{h(c)}\right)^{1/e} \mod n$, if e is odd, and if $|e|_2 = l_e$.

Unfortunately, the proof of the more general chameleon hash scheme class does not formally transfer to the Cramer-Shoup signature scheme because in the Cramer-Shoup scheme the key material of its chameleon hash function is not chosen independently. In particular, the chameleon hash function uses the same RSA modulus and the same value v. This requires slightly more care in the security proof. We provide a full proof of the Cramer-Shoup signature scheme in Section 4.3.5.

4.3 Security

Theorem 4.3 *The Cramer-Shoup signature scheme, the combining signature class (in both the SRSA and the SDH setting), and the chameleon signature class (in both the SRSA and the SDH setting) are tightly secure against adaptive chosen message attacks. In particular, this implies that the Camenisch-Lysyanskaya, the Fischlin, the Zhu, and the SDH based Camenisch-Lysyanskaya scheme are tightly secure against strong existential forgeries under adaptive chosen message attacks.*

We subsequently provide the intuition behind our security proofs. We also show how to transfer our technique to $\mathcal{S}_{\mathrm{CH}}$. In Section 4.3.4, we present a *full* proof of security for $\mathcal{S}_{\mathrm{CMB,SRSA}}$, which seems to us to be the technically most involved reduction. The proof of $\mathcal{S}_{\mathrm{CMB,SDH}}$ proceeds analogously and appears in Section 4.3.6. In Section 4.3.5 we provide a full proof of security of the Cramer-Shoup signature scheme.

4.3.1 The SRSA Based Schemes

Let us first consider the SRSA based schemes, where $\mathcal{B}$ is given an SRSA challenge $(\hat{u}, n)$ with $\hat{u} \in \mathbb{Z}_n^*$. Assume that attacker $\mathcal{A}$ issues q signature queries $m_1, \ldots, m_q \in \mathcal{M}$. As a response to each query m_i with $i \in [1; q]$, $\mathcal{A}$ receives a corresponding signature $\sigma_i = (r_i, s_i, e_i) \in \mathcal{R} \times QR_n \times E$. Recall that the existing security proofs for schemes of the combining class (e.g. [54]) consider two forgers that loosely reduce from the SRSA assumption. This is the case when it holds for $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$ that $\gcd(e^*, \prod_{i=1}^{q} e_i) \neq 1$.[4] Given that $|e^*|_2 = l_e$ this means that $e^* = e_j$ for some $j \in [1; q]$. Let us concentrate on the case that $r^* \neq r_j$. The proof of the remaining case ($e^* = e_j$, $r^* = r_j$ and $m^* \neq m_j$) is very similar. It additionally exploits the properties of the combining function. The proofs

[4]The proof of the case $\gcd(e^*, \prod_{i=1}^{q} e_i) = 1$ is straight-forward.

in [45, 54, 103, 30, 104] work as follows: the simulator $\mathcal{B}$ at first guesses $j \overset{\$}{\leftarrow} \{1, \ldots, q\}$. By construction, $\mathcal{B}$ can answer all signature queries but only if $\mathcal{A}$ outputs a forgery where $e^* = e_j$ it can extract a solution to the SRSA challenge. In all other cases (if $e^* = e_i$ for some $i \in \{1, \ldots, q\} \backslash \{j\}$), $\mathcal{B}$ just aborts. Since the number of signature queries q rises polynomially in the security parameter, the probability for $\mathcal{B}$ to correctly guess j in advance is q^{-1} and thus not negligible. However, the security reduction loses a factor of q here. Our aim is to improve this reduction step. Ideally, we have that *any* forgery which contains $e^* \in \{e_1, \ldots, e_q\}$ helps the simulator to break the SRSA assumption. As a result, the simulator can completely avoid guessing. The main task is to re-design the way $\mathcal{B}$ computes $\mathcal{A}$'s input parameters: for *every* $i \in \{1, \ldots, q\}$, we must have exactly one choice of r_i such that $\mathcal{B}$ can simulate the signing oracle without having to break the SRSA challenge. On the other hand, if $\mathcal{A}$ outputs $(m^*, (r^*, s^*, e^*))$ with $e^* = e_i$ for some $i \in [1; q]$ and $r^* \neq r_i$, $\mathcal{B}$ must be able to compute a solution to the SRSA challenge. Let us now go into more detail.

For simplicity, assume that $\mathcal{B}$ can setup $\mathcal{A}$'s input parameters such that the verification of a signature $\sigma = (r, s, e)$ always reduces to

$$s^e = \hat{u}^{f(r)} \bmod n. \tag{4.1}$$

Suppose that neither $\hat{u}$ nor $f : \mathcal{R} \rightarrow \mathbb{N}$ are ever revealed to $\mathcal{A}$. We exploit that the r_i are chosen independently at random. So, they can be specified *prior* to the signature queries. Now, $\mathcal{B}$'s strategy to simulate the signing oracle is to define $r_1, \ldots, r_q$ such that for every $i \in [1; q]$ it can compute a prime $e_i \in E$ with $e_i | f(r_i)$. Without having to break the SRSA assumption, $\mathcal{B}$ can then compute $s_i = \hat{u}^{f(r_i)/e_i}$ and output the i-th signature as (r_i, s_i, e_i). Let us now turn our attention to the extraction phase where $\mathcal{B}$ is given $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption we have $e^* = e_i$ for some $i \in [1; q]$ and $r^* \neq r_i$. $\mathcal{B}$ wants to have that $\gcd(e^*, f(r^*)) = D < e^*$ (or $f(r^*) \neq 0 \bmod e^*$) because then it can find a solution to the SRSA challenge by computing $a, b \in \mathbb{Z} \setminus \{0\}$ with $af(r^*)/D + be^*/D = 1$ using extended Euclidean algorithm and outputting

$$(s^*)^a \hat{u}^b = \hat{u}^{D/e^*}, e^*/D.$$

$\mathcal{B}$'s strategy to guarantee $\gcd(e^*, f(r^*)) = D < e^*$ is to ensure that $e^* = e_i \not| f(r^*)$. Unfortunately, $\mathcal{B}$ cannot foresee r^*. Therefore, the best solution is to design f such that $e_i \not| f(r^*)$ for *all* $r^* \neq r_i$.

Obviously, $\mathcal{B}$ makes strong demands on $f(r)$. We will now present the basic idea that stand behind our construction of $f(r)$.

Design Philosophy. Recall polynomial interpolation. Given two sets $A = \{a_1, \ldots, a_q\} \subseteq \mathbb{N}$ and $B = \{b_1, \ldots, b_q\} \subseteq \mathbb{N}$ one can easily design a polynomial f of maximal degree q with the following property: for every $i \in [1; q]$ it holds that

$$c = a_i \Rightarrow f(c) = b_i.$$

In the following we want to modify the properties of this function according to the requirements of our security reduction in Section 4.3. First, f should be linear such that it can efficiently be embedded in the exponents of only two group elements. Second, for f to be useful in the simulation phase *and* the extraction phase of the security reduction we need equivalence instead of implication. Of course, we can only buy these additional properties by relaxing others: if we consider the definition of the SRSA assumption (and Equation 4.1), it is perfectly sufficient to just require $b_i | f(c)$ instead of $f(c) = b_i$. In the simulation phase this immediately transfers to the situation where the simulator has to pretend that it can compute solutions to the SRSA problem. On the other hand, if $c \neq a_i$ we get (by the required equivalence) that also $b_i \not| f(c)$ what directly gives rise to a solution to the SRSA problem.

Lemma 4.4 *Suppose we are given a set of q primes $P = \{e_1, \ldots, e_q\}$ with $2^{l_e-1} \leq e_i \leq 2^{l_e}-1$ for $i \in [1;q]$ and q integers $c_1, \ldots, c_q \in [0; 2^{l_e-1}-1]$ ($|c_i|_2 < l_e$ for all $i \in [1;q]$). Then we can construct an efficient* linear *function $f : [0; 2^{l_e-1}-1] \rightarrow \mathbb{Z}$ such that*

$$e_k | f(c) \Leftrightarrow c = c_k$$

for all $k \in [1;q]$.

Proof. Let

$$f(c) := \sum_{i=1}^{q} c_i \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j - c \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j.$$

First assume $c = c_k$. Then $f(c)$ reduces to

$$f(c_k) = \sum_{\substack{i=1 \\ i \neq k}}^{q} (c_i - c_k) \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j = e_k \sum_{\substack{i=1 \\ i \neq k}}^{q} c_i \prod_{\substack{j=1 \\ j \neq i,k}}^{q} e_j = 0 \bmod e_k.$$

Now assume $c \neq c_k$. Then we have

$$\begin{aligned} f(c) &= \sum_{i=1}^{q} (c_i - c) \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j = \sum_{\substack{i=1 \\ i \neq k}}^{q} (c_i - c) \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j + (c_k - c) \prod_{\substack{j=1 \\ j \neq k}}^{q} e_j \\ &= e_k \sum_{\substack{i=1 \\ i \neq k}}^{q} (c_i - c) \prod_{\substack{j=1 \\ j \neq i,k}}^{q} e_j + (c_k - c) \prod_{\substack{j=1 \\ j \neq k}}^{q} e_j \\ &= (c_k - c) \prod_{\substack{j=1 \\ j \neq k}}^{q} e_j \bmod e_k. \end{aligned}$$

Since the e_i are distinct primes and as $|c_k - c| < e_k$, it holds that $(c_k - c) \prod_{j=1, j \neq k}^{q} e_j \neq 0 \bmod e_k$ what proves Lemma 4.4. If we assume that $|c_i| < l_e + 1$ for all $i \in [1;q]$ we can also have $c_1, \ldots, c_q \in [-2^{l_e}; 2^{l_e}-1]$. □

We can now directly use Lemma 4.4 and define $f(r)$ as

$$f(r) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1\\j\neq i}}^{q} e_j - r \sum_{i=1}^{q} \prod_{\substack{j=1\\j\neq i}}^{q} e_j, \tag{4.2}$$

for $r_1, \ldots, r_q \in \mathcal{R}$. Observe that we must guarantee in the proof of security that the $e_1, \ldots, e_q \in E$ are distinct primes. As the prime numbers are chosen at random from a large set $|E|$ in the signature generation, a union bound can easily show that collisions occur only with negligible probability.

4.3.2 The SDH Based Schemes

Under the SDH assumption, the situation is very similar. Here we also analyze three possible types of forgeries $(m^*, (r^*, s^*, t^*))$: 1.) $t^* \notin \{t_1, \ldots, t_q\}$, 2.) $t^* = t_i$ with $i \in [1; q]$ but $r^* \neq r_i$, and 3.) $t^* = t_i$, $r^* = r_i$ (but $m^* \neq m_i$) with $i \in [1; q]$. Again, we concentrate on the second case. At the beginning, $\mathcal{B}$ is given an SDH challenge $\left(\hat{g}_1, \hat{g}_1^x, \hat{g}_1^{(x^2)}, \ldots, \hat{g}_1^{(x^q)}, g_2, g_2^x\right)$. This time, $\mathcal{B}$ chooses $PK = g_2^x$. In the SDH setting, Equation (4.1) transfers to

$$e(s, PK g_2^t) = e(\hat{g}_1^{f(r,x)}, g_2) \Leftrightarrow s^{x+t} = \hat{g}_1^{f(r,x)}. \tag{4.3}$$

In contrast to the SRSA setting, f is now a polynomial with indeterminate x and maximal degree q. Again, $\mathcal{B}$ must keep $f(r, x)$ and the $\hat{g}_1^{(x^i)}$ secret from $\mathcal{A}$. We define

$$f(r, x) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1\\j\neq i}}^{q} (x + t_j) - r \sum_{i=1}^{q} \prod_{\substack{j=1\\j\neq i}}^{q} (x + t_j),$$

for $r_1, \ldots, r_q \in \mathcal{R}$ and distinct $t_1, \ldots, t_q \in \mathbb{Z}_p$. Using the SDH challenge, $\mathcal{B}$ can easily compute $\hat{g}_1^{f(r,x)}$ since $f(r, x)$ has maximal degree q. The following lemma shows that $f(r, x)$ indeed has the required properties.

Lemma 4.5 *Given a set $T = \{t_1, \ldots, t_q\} \subseteq \mathbb{Z}_p$ and q values $c_1, \ldots, c_q \in \mathbb{Z}_p$, we can easily build a linear (in c) function $f(c, x)$ with $f : \mathbb{Z}_p \times \mathbb{Z}_p[x] \to \mathbb{Z}_p[x]$ that maps polynomials of maximal degree q in indeterminate x to polynomials of maximal degree q such that*

$$(x + t_k) | f(c, x) \Leftrightarrow c = c_k$$

for all $k \in [1; q]$.

Proof. Let

$$f(c, x) := \sum_{i=1}^{q} c_i \prod_{\substack{j=1\\j\neq i}}^{q} (x + t_j) - c \sum_{i=1}^{q} \prod_{\substack{j=1\\j\neq i}}^{q} (x + t_j).$$

First assume $c = c_k$. Then $f(c, x)$ reduces to

$$f(c_k, x) = \sum_{\substack{i=1\\i\neq k}}^{q}(c_i - c)\prod_{\substack{j=1\\j\neq i}}^{q}(x + t_j) = (x + t_k)\sum_{\substack{i=1\\i\neq k}}^{q}(c_i - c)\prod_{\substack{j=1\\j\neq i,k}}^{q}(x + t_j).$$

Now assume $c \neq c_k$. Then we have

$$\begin{aligned} f(c, x) &= \sum_{i=1}^{q}(c_i - c)\prod_{\substack{j=1\\j\neq i}}^{q}(x + t_j) \\ &= \sum_{\substack{i=1\\i\neq k}}^{q}(c_i - c)\prod_{\substack{j=1\\j\neq i}}^{q}(x + t_j) + (c_k - c)\prod_{\substack{j=1\\j\neq k}}^{q}(x + t_j) \\ &= (x + t_k)\sum_{\substack{i=1\\i\neq k}}^{q}(c_i - c)\prod_{\substack{j=1\\j\neq i,k}}^{q}(x + t_j) + (c_k - c)\prod_{\substack{j=1\\j\neq k}}^{q}(x + t_j). \end{aligned}$$

Since the t_i are distinct it holds that $(x+t_k) \not| (c_k-c)\prod_{j=1,j\neq k}^{q}(x+t_j)$ what proves Lemma 4.5.
$\square$

Similar to before, to apply Lemma 4.5 we must guarantee that the t_i are all distinct. Since the t_i are chosen from $\mathbb{Z}_p$, this probability is negligible.

4.3.3 Security of the Chameleon Hash Signature Class

The chameleon hash class is also tightly secure in the SRSA and the SDH setting. For convenience let $c_i = ch(r_i, m_i)$ for $i \in [1; q]$ and $c^* = ch(r^*, m^*)$. Altogether there are again three types of forgeries to consider: 1) $e^* \notin \{e_1, \ldots, e_q\}$ ($t^* \notin \{t_1, \ldots, t_q\}$), 2) $e^* = e_i$ ($t^* = t_i$) but $c^* \neq c_i$, and 3) $e^* = e_i$ ($t^* = t_i$), $c^* = c_i$ but $m^* \neq m_i$. The proof of 1) is straight-forward and very similar to the proof of Type I forgers of the combining class. The proof of 3) clearly reduces to the security properties of the chameleon hash function. The proof of 2) requires our new technique to set up $f(c)$ ($f(c, x)$) (similar to the proof of the Cramer-Shoup signature scheme in Section 4.3.5). Recall, that the proof of the combining class concentrates on the equations $s^e = \hat{u}^{f(c)}$ and $f(c) = \sum_{i=1}^{q} c_i \prod_{j=1,j\neq i}^{q} e_j - c\sum_{i=1}^{q}\prod_{j=1,j\neq i}^{q} e_j$ in the SRSA setting (and $s^{x+t} = \hat{g}_1^{f(c,x)}$ and $f(c, x) = \sum_{i=1}^{q} c_i \prod_{j=1,j\neq i}^{q}(x + t_j) - c\sum_{i=1}^{q}\prod_{j=1,j\neq i}^{q}(x + t_j)$ in the SDH setting). In the proof of the combining class the c_i are random values ($c_i = r_i$) that can be specified prior to the simulation phase. In the proof of the chameleon hash class we take a similar approach. Now the c_i are the output values of a chameleon hash function. In the initialization phase of the proof we choose q random input pairs $({m'}_i, {r'}_i) \in \mathcal{M} \times \mathcal{R}$, $i \in [1; q]$ to compute the $c_i = \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m'_i, r'_i)$. Then we prepare the function $f(c)$ ($f(c, x)$) with $C = \{c_1, \ldots, c_q\}$ and a set of q random primes l_e-bit primes (random values $t_1, \ldots, t_q \in \mathbb{Z}_p$) as in the proofs of the combining class using Lemma 4.4 (Lemma 4.5). Next, we embed

$f(c)$ $(f(c,x))$ in the exponents of the two group elements u, v (a, b). In the simulation phase we give the simulator SK_{CH} to map the attacker's messages m_i to the prepared c_i by computing $r_i = \mathsf{CH.Coll}(SK_{\mathsf{CH}}, r'_i, m'_i, m_i)$. In this way we can successfully simulate the signing oracle. In the extraction phase, the properties of the chameleon hash function guarantee that $c^* \notin \{c_1, \ldots, c_q\}$ (otherwise we can break the security of the chameleon hash function). This ensures that we can find a solution to the SRSA challenge (SDH challenge).

4.3.4 Security Analysis of $\mathcal{S}_{\mathbf{CMB,SRSA}}$

Lemma 4.6 *Assume we work in the SRSA setting such that the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds and $\mathcal{V}$ is a $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining function. Then, the combining signature class as presented in Section 4.2.1 is (q, t, ϵ)-secure*[5] *against adaptive chosen message attacks provided that*

$$\epsilon \leq \frac{9}{2}\epsilon_{SRSA} + 3\epsilon_{comb} + 3q\delta_{comb} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}, \quad t = t_{SRSA} - \mathsf{T}_{SRSA,comb}(q^2).$$

The proof of Lemma 4.6 is the first step in the proof of Theorem 4.3. It implies that the original Camenisch-Lysyanskaya, the Fischlin and the Zhu's signature scheme are tightly secure against existential forgeries under adaptive chosen message attacks. Proof. Assume that $\mathcal{A}$ is a forger that (q, t, ϵ)-breaks the strong existential unforgeability of $\mathcal{S}_{\mathrm{CMB,SRSA}}$. Then, we can construct a simulator $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the SRSA problem in time t_{SRSA} with advantage ϵ_{SRSA}. We consider three types of forgers that after q queries $m_1, \ldots, m_q$ and corresponding responses $(r_1, s_1, e_1), \ldots, (r_q, s_q, e_q)$ partition the set of all possible forgeries $(m^*, (r^*, s^*, e^*))$. In the proof, we treat all types of attackers differently. At the beginning, we let $\mathcal{B}$ guess with probability at least $\frac{1}{3}$ which forgery $\mathcal{A}$ outputs. Lemma 4.6 then follows by a standard hybrid argument. We assume that $\mathcal{B}$ is given an SRSA challenge instance $(\hat{u}, n)$. Let $\Pr[S_i]$ denote the success probability of an attacker to successfully forge signatures in Game i.

Type I Forger $(e^* \notin \{e_1, \ldots, e_q\})$
Suppose $\mathcal{B}$ guesses that $\mathcal{A}$ is a Type I Forger.
Game$_0$. This is the original attack game. By assumption, $\mathcal{A}$ (q, t, ϵ)-breaks $\mathsf{SIG}_{\mathrm{CMB,SRSA}}$ when interacting with the signing oracle $\mathcal{O}(SK, \cdot)$. We have that,

$$\Pr[S_0] = \epsilon. \tag{4.4}$$

Game$_1$. Now, $\mathcal{B}$ constructs the values u, v, w using the SRSA challenge instead of choosing them randomly from QR_n. First, $\mathcal{B}$ chooses q random primes $e_1, \ldots, e_q \overset{\$}{\leftarrow} E$ and three random elements $t'_0, t''_0 \overset{\$}{\leftarrow} \mathbb{Z}_{(n-1)/4}$ and $t_0 \overset{\$}{\leftarrow} \mathbb{Z}_{3(n-1)/4}$. In the following let $\bar{e} := \prod_{k=1}^{q} e_k$, $\bar{e}_i := \prod_{k=1,k\neq i}^{q} e_k$ and $\bar{e}_{i,j} := \prod_{k=1,k\neq i,k\neq j}^{q} e_k$. The simulator computes $u = \hat{u}^{2t_0\bar{e}}$, $v =$

[5]Using explicit bounds on the prime counting function [93], we can lower bound the number of primes in E for $l_e \geq 7$ as $|E| > (2^{l_e} - 1)/(\ln(2^{l_e} - 1) + 2) - (2^{l_e-1} - 1)/(\ln(2^{l_e-1} - 1) - 4)$.

$\hat{u}^{2t'_0\bar{e}}$, $w = \hat{u}^{2t''_0\bar{e}}$ using the SRSA challenge. Since the t_0, t'_0, t''_0 are not chosen uniformly at random from $\mathbb{Z}_{p'q'}$ we must analyze the success probability for $\mathcal{A}$ to detect our construction. Observe that $(n-1)/4 = p'q' + (p'+q')/2 > p'q'$. Without loss of generality let $p' > q'$. Now, the probability of a randomly chosen $x \in \mathbb{Z}_{(n-1)/4}$ not to be in $\mathbb{Z}_{p'q'}$ is

$$\Pr[x \xleftarrow{\$} \mathbb{Z}_{(n-1)/4},\ x \notin \mathbb{Z}_{p'q'}] = 1 - \frac{|\mathbb{Z}_{p'q'}|}{|\mathbb{Z}_{(n-1)/4}|} = \frac{(p'+q')}{(2p'q'+p'+q')} < \frac{1}{q'+1} < 2^{-(|q'|_2-1)}.$$

With the same arguments we can show that t_0 is also distributed almost uniformly at random in $\mathbb{Z}_{p'q'}$ and $\mathbb{Z}_{3p'q'}$. Since the e_i are primes smaller than p' and q' it holds that $e_i \not| p'q'$. Therefore, the distribution of the generators is almost equal to the previous game and we get by a union bound that

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)}\ . \tag{4.5}$$

Game$_2$. Now, $\mathcal{B}$ simulates $\mathcal{O}(SK, \cdot)$ by answering $\mathcal{A}$'s signature queries. Subsequently, set $z_j = z(e_j, m_j)$ and $z^* = z(e^*, m^*)$. The simulator $\mathcal{B}$ sets $PK = n$ and for all $j \in \{1, \ldots, q\}$ it chooses a random $r_j \in \mathcal{R}$ and outputs $\sigma_j = (r_j, s_j, e_j)$ with $s_j = (uv^{r_j}w^{z_j})^{\frac{1}{e_j}} = \hat{u}^{2(t_0+t'_0r_j+t''_0z_j)\bar{e}_j}$. The distribution of the so computed values is equal to the previous game and

$$\Pr[S_2] = \Pr[S_1]\ . \tag{4.6}$$

Game$_3$. Now, consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. Define $\hat{e} = (t_0 + t'_0r^* + t''_0z^*)$. For $\mathcal{A}$'s forgery it holds that $(s^*)^{e^*} = \hat{u}^{2\bar{e}\hat{e}}$. We also have that $\gcd(e^*, 2\bar{e}\hat{e}) = \gcd(e^*, \hat{e})$ since by assumption we know that $\gcd(e^*, 2\bar{e}) = 1$. We will now analyze the probability for the event $\gcd(e^*, \hat{e}) < e^*$ to happen. If $\gcd(e^*, \hat{e}) = e^*$ (or $\hat{e} = 0 \bmod e^*$) $\mathcal{B}$ simply aborts and restarts. Since $|e^*|_2 = l_e$, it holds that $\gcd(e^*, p'q') < e^*$. Write $t_0 \in \mathbb{Z}_{3(n-1)/4}$ as $t_0 = t_{0,1} + p'q't_{0,2}$ where $t_{0,2} \in [0;2]$ and $t_{0,1} \in [0, p'q'-1]$ and observe that $\mathcal{A}$'s view is independent from $t_{0,2}$. Let $T = \hat{e} - p'q't_{0,2}$. We now argue that there exists at most one $\tilde{t}_{0,2} \in [0;2]$ such that $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$. This is crucial because if $\mathcal{A}$ produces forgeries with $T + \tilde{t}_{0,2}p'q' = 0 \bmod e^*$ for *all* $\tilde{t}_{0,2} \in [0;2]$ it always holds that $\gcd(e^*, \hat{e}) = e^*$ and $\mathcal{B}$ cannot extract a solution the the SRSA challenge (using the techniques described below).
Assume there exists at least one such $\tilde{t}_{0,2}$. Then, we have that $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$. Let us analyze the remaining possibilities $\tilde{t}_{0,2} \pm 1$ and $\tilde{t}_{0,2} \pm 2$ as $A = T + \tilde{t}_{0,2}p'q' \pm p'q' \mod e^*$ and $B = T + \tilde{t}_{0,2}p'q' \pm 2p'q' \mod e^*$. Since $\gcd(e^*, p'q') < e^*$ we know that $p'q' \neq 0 \bmod e^*$. As $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$ we must have that $A \neq 0 \bmod e^*$. Also, because e^* is odd we know that $2p'q' \neq 0 \bmod e^*$ and thus $B \neq 0 \bmod e^*$. So, because there can only exist at most one $\tilde{t}_{0,2} \in [0;2]$ with $\gcd(e^*, \hat{e}) = e^*$ and since this $\tilde{t}_{0,2}$ is hidden from $\mathcal{A}$'s view, $\mathcal{A}$'s probability to output it is at most 1/3. This means that with probability at least 2/3, $\mathcal{B}$ has that $\gcd(e^*, \hat{e}) = d < e^*$. From $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$, $\mathcal{B}$ can now find a solution to the SRSA challenge by computing $a, b \in \mathbb{Z}$ with $\gcd(e^*/d, 2\bar{e}\hat{e}/d) = ae^*/d + b2\bar{e}\hat{e}/d = 1$ and

$$\hat{u}^{d/e^*} = \hat{u}^a(s^*)^b, e^*/d.$$

Finally, we have that

$$\Pr[S_3] \geq 2 \cdot \Pr[S_2]/3 \tag{4.7}$$

and

$$\Pr[S_3] = \epsilon_{\text{SRSA}} \ . \tag{4.8}$$

Plugging in Equations (4.4)–(4.8), we get that $\epsilon \leq \frac{3}{2}\epsilon_{\text{SRSA}} + 3 \cdot 2^{2-l_n/2}$.

Type II Forger $(e^* = e_i$ and $r^* \neq r_i)$
Now suppose $\mathcal{B}$ expects $\mathcal{A}$ to be a Type II Forger. We only present the differences to the previous proof.

Game$_1$. First, $\mathcal{B}$ randomly chooses q *distinct* l_e-bit primes $e_1, \ldots, e_q$ and q random elements $r_1, \ldots, r_q \in \mathcal{R}$. Additionally, it chooses three random elements t_0, t_0', t_0'' from $\mathbb{Z}_{(n-1)/4}$. Next, $\mathcal{B}$ computes $u = \hat{u}^{2\left(t_0\bar{e}+\sum_{i=1}^{q} r_i\bar{e}_i\right)}$, $v = \hat{u}^{2\left(t_0'\bar{e}-\sum_{i=1}^{q}\bar{e}_i\right)}$, and $w = \hat{u}^{2t_0''\bar{e}}$ using the SRSA challenge. Again,

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)} \ . \tag{4.9}$$

Game$_2$. Now $\mathcal{B}$ simulates the signing oracle $\mathcal{O}(SK, \cdot)$. On each signature query m_j with $j \in \{1, \ldots, q\}$, $\mathcal{B}$ responds with $\sigma_j = (r_j, s_j, e_j)$ using the precomputed r_j and e_j and computing s_j as

$$s_j = \left(uv^{r_j}w^{z_j}\right)^{\frac{1}{e_j}} = \hat{u}^{2\left((t_0+t_0'r_j+t_0''z_j)\bar{e}_j+\sum_{i=1}^{q} r_i\bar{e}_{i,j}-r_j\sum_{i=1}^{q}\bar{e}_{i,j}\right)} = \hat{u}^{2\left((t_0+t_0'r_j+t_0''z_j)\bar{e}_j+\sum_{i=1,i\neq j}^{q}(r_i-r_j)\bar{e}_{i,j}\right)} \ .$$

Since we have chosen the e_i to be distinct primes we have by a union bound that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|} \ . \tag{4.10}$$

Game$_3$. Now consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption there is a $i \in \{1, \ldots, q\}$ with $e^* = e_i$ and $r_i \neq r^*$. Then we have that

$$\left((s^*) \cdot \hat{u}^{-2\left((t_0+t_0'r^*+t_0''z^*)\bar{e}_i+\sum_{j=1,j\neq i}^{q}(r_j-r^*)\bar{e}_{i,j}\right)}\right)^{e_i} = \hat{u}^{2(r_i-r^*)\bar{e}_i}.$$

Since $|r_i - r^*| < e_i$ and e_i is an odd prime we have that $\gcd(2(r_i - r^*), e_i) = 1$ and as before we can compute $\hat{u}^{\frac{1}{e_i}}$ which is a solution to the SRSA challenge.

$$\Pr[S_3] = \epsilon_{\text{SRSA}} \ . \tag{4.11}$$

Summing up Equations (4.9)–(4.11), we get that $\epsilon \leq \epsilon_{\text{SRSA}} + q^2/|E| + 3 \cdot 2^{2-l_n/2}$.

Type III Forger $(e^* = e_i$ and $r^* = r_i)$
In case $\mathcal{B}$ expects $\mathcal{A}$ to be a Type III Forger, there are only minor differences as compared to the previous proof.

Game$_1$. First, $\mathcal{B}$ randomly chooses q l_e-bit primes $e_1, \ldots, e_q$ and q random $z_1, \ldots, z_q \in \mathcal{Z}$. Then, $\mathcal{B}$ draws three random elements t_0, t_0', t_0'' from $\mathbb{Z}_{(n-1)/4}$. Next, $\mathcal{B}$ computes u, v, and w as $u = \hat{u}^{2\left(t_0\bar{e}+\sum_{i=1}^{q} z_i\bar{e}_i\right)}$, $v = \hat{u}^{2t_0'\bar{e}}$, and $w = \hat{u}^{2\left(t_0''\bar{e}-\sum_{i=1}^{q}\bar{e}_i\right)}$.

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)} \ . \tag{4.12}$$

Game$_2$. This game is equal to the previous game except that we require the e_i to be all distinct. We have that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|} \; . \tag{4.13}$$

Game$_3$. Now $\mathcal{B}$ simulates the signing oracle. For each query m_j with $j \in \{1, \ldots, q\}$, $\mathcal{B}$ computes $r_j = z^{-1}(z_j, m_j)$. If $r_j \notin \mathcal{R}$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs the signature $\sigma_j = (r_j, s_j, e_j)$ with s_j being computed as

$$s_j = \left(uv^{r_j}w^{z_j}\right)^{\frac{1}{e_j}} = \hat{u}^{2\left((t_0+t_0'r_j+t_0''z_j)\bar{e}_j+\sum_{i=1}^{q} z_i\bar{e}_{i,j}-z_j\sum_{i=1}^{q}\bar{e}_{i,j}\right)} = \hat{u}^{2\left((t_0+t_0'r_j+t_0''z_j)\bar{e}_j+\sum_{i=1,i\neq j}^{q}(z_i-z_j)\bar{e}_{i,j}\right)} \; .$$

The properties of the combining function guarantee that the r_j are statistically close to uniform over $\mathcal{R}$ such that,

$$\Pr[S_3] \geq \Pr[S_2] - q\delta_{\text{comb}} \; . \tag{4.14}$$

Game$_4$. This game is like the previous one except that $\mathcal{B}$ aborts whenever there is a collision such that $z_i = z(r_i, m_i) = z(r_i, m^*) = z^*$ for some r_i. Observe that we must have $m^* \neq m_i$, otherwise $\mathcal{A}$ just replayed the i-the message/signature pair. For all t_{comb}-time attackers this happens with probability at most ϵ_{comb}. Therefore,

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\text{comb}} \; . \tag{4.15}$$

Consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption, there is one index $i \in \{1, \ldots, q\}$ with $e^* = e_i$ and $r^* = r_i$. For this index it holds that

$$\left((s^*) \cdot \hat{u}^{-2\left((t_0+t_0'r^*+t_0''z^*)\bar{e}_i+\sum_{j=1,j\neq i}^{q}(z_j-z^*)\bar{e}_{i,j}\right)}\right)^{e_i} \;=\; \hat{u}^{2(z_i-z^*)\bar{e}_i} \; .$$

Since we have excluded collisions, it follows that $z_i \neq z^*$. As $|z_i - z^*| \leq e_i$, $\mathcal{B}$ can compute $\hat{u}^{\frac{1}{e_i}}$ as a solution to the SRSA challenge. Finally,

$$\Pr[S_4] = \epsilon_{\text{SRSA}} \; . \tag{4.16}$$

Summing up Equations (4.12)–(4.16), we get that $\epsilon \leq \epsilon_{\text{SRSA}}+\epsilon_{\text{comb}}+q\delta_{\text{comb}}+q^2/|E|+3\cdot 2^{2-l_n/2}$.

Clearly, the running time of $\mathcal{B}$ is $t + \mathsf{T}_{\text{SRSA,comb}}(q^2)$ where $\mathsf{T}_{\text{SRSA,comb}}(q^2)$ denotes the additional time to compute u, v, w, the signatures in the simulation phase, and the solution to the SRSA problem. $\mathsf{T}_{\text{SRSA,comb}}(q^2)$ is dominated by q^2 exponentiations in $\mathbb{Z}_n^*$. $\square$

4.3.5 Security Analysis of the Cramer-Shoup Signature Scheme

The original proof in [45] considers three types of forgers that after q queries $m_1, \ldots, m_q$ and corresponding responses $(r_1, s_1, e_1), \ldots, (r_q, s_q, e_q)$ partition the set of all possible forgeries $(m^*, (r^*, s^*, e^*))$. For the following two forgers the original proof already tightly reduces to the SRSA assumption.

For a **Type I Forger** it holds that $e^* \notin \{e_1, \ldots, e_q\}$. The proof is very similar to the proof of Type I Forgers in the combining class.

A **Type II Forger** outputs a forgery such that $e^* = e_j$ for some $j \in [1;q]$. It holds that $c^* = (r^*)^{\tilde{e}} v^{h(m^*)} = (r_j)^{\tilde{e}} v^{h(m_j)} = c_j \bmod n$. This proof reduces security from the implicit chameleon hash function and the hash function. ($c^* = c_j$ with $m^* \neq m_j$ constitutes a collision in the chameleon hash function or a hash collision.)

The only loose reduction is the proof of **Type III Forgers**.

A **Type III Forger** outputs a forgery with $e^* = e_j$ for some $j \in [1;q]$. It holds that $c^* = (r^*)^{\tilde{e}} v^{h(m^*)} \neq (r_j)^{\tilde{e}} v^{h(m_j)} = c_j \bmod n$. We will now present a new reduction for **Type III Forgers** that makes use of the technique described in Section 4.3.1.

We assume that $\mathcal{B}$ is given an SRSA challenge instance $(\hat{u}, n)$. Intuitively, we must only give a new proof for the case that the adversary outputs a forgery with $e^* = e_j$ for some $j \in [1;q]$ and $c^* = (r^*)^{\tilde{e}} v^{h(m^*)} \neq (r_j)^{\tilde{e}} v^{h(m_j)} = c_j \bmod n$. Recall (4.1) and (4.2). In our new proof the r_i now correspond to the output values of the implicit chameleon hash function $ch(r,m) = r^{\tilde{e}} v^{h(m)} \bmod n$. Nevertheless these output values, as well as the e_i, can also be specified already in the initialization phase. This is because the simulator $\mathcal{B}$ is given the secret key of the chameleon hash function what enables him to map the adversary's messages to the prespecified values. Since $f(r)$ is linear, $\mathcal{B}$ can efficiently be embedded in the exponents of u, v. By assumption the adversary outputs a forgery that maps to a new output value of the chameleon hash function ($c^* = (r^*)^{\tilde{e}} v^{h(m^*)} \neq (r_j)^{\tilde{e}} v^{h(m_j)} = c_j \bmod n$). As before we can use it to extract a solution to the SRSA assumption.

Lemma 4.7 *Assume the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds. Moreover let $h : \{0,1\}^* \rightarrow \{0,1\}^{l_c}$ be a $(t_h = t_{SRSA}, \epsilon_h)$-collision-resistant hash function. Then, the Cramer-Shoup signature scheme is (q, t, ϵ)-secure against Type III forgers provided that*

$$q = q_{SRSA}, \quad \epsilon \leq \epsilon_{SRSA} + \epsilon_h + \frac{q^2}{|E|} + 2^{3-l_n/2}, \quad t \approx t_{SRSA} - \mathsf{T}_{SDH,comb}.$$

Proof.

Game$_0$. This is the original attack game. By assumption, $\mathcal{A}$ (q,t,ϵ)-breaks $\mathsf{SIG}_{\mathrm{CMB,SRSA}}$ when interacting with the signing oracle $\mathcal{O}(SK, \cdot)$. We have that,

$$\Pr[S_0] = \epsilon. \tag{4.17}$$

Game$_1$. Now, $\mathcal{B}$ constructs the values u, v using the SRSA challenge instead of choosing them randomly from QR_n. First, $\mathcal{B}$ chooses q random primes $e_1, \ldots, e_q \xleftarrow{\$} E$ and two random elements $t_0, t_0' \xleftarrow{\$} \mathbb{Z}_{(n-1)/4}$. Let again $\bar{e} := \prod_{k=1}^{q} e_k$, $\bar{e}_i := \prod_{k=1, k\neq i}^{q} e_k$ and $\bar{e}_{i,j} := \prod_{k=1, k\neq i, k\neq j}^{q} e_k$. Then $\mathcal{B}$ chooses q random values $c_1', \ldots, c_q' \xleftarrow{\$} QR_n$. Next it chooses $\tilde{e} \xleftarrow{\$} E$ such that $\tilde{e} \notin \{e_1, \ldots, e_q\}$. For all $i \in [1;q]$ it then computes $c_i = (c_i')^{\tilde{e}} \bmod n$. The values u, v are computed as $v = \hat{u}^{-2\tilde{e}(t_0\bar{e} + \sum_{i=1}^{q} \bar{e}_i)}$ and $u = \hat{u}^{2\tilde{e}(t_0'\bar{e} + \sum_{i=1}^{q} h(c_i)\bar{e}_i)}$. For convenience let

$v' = \hat{u}^{-2(t'_0\bar{e}+\sum_{i=1}^{q}\bar{e}_i)}$ and observe that $(v')^{\tilde{e}} = v \bmod n$. The distribution of the generators is almost equal to the previous game and we get by a union bound that

$$\Pr[S_1] \geq \Pr[S_0] - 2 \cdot 2^{-(l_n/2-2)} \ . \tag{4.18}$$

Game$_2$. In this game the simulator aborts if there is a collision among the randomly chosen e_i. We have that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|} \ . \tag{4.19}$$

Game$_3$. Now, $\mathcal{B}$ simulates $\mathcal{O}(SK,\cdot)$ by answering $\mathcal{A}$'s signature queries. For all $j \in \{1,\ldots,q\}$ it first computes $r_j \in QR_n$ as $r_j = c'_j(v')^{h(m_j)} \bmod n$. Observe that by construction $r_j^{\tilde{e}}/v^{h(m_j)} = c_j \bmod n$. Next $\mathcal{B}$ outputs $\sigma_j = (r_j, s_j, e_j)$. The distribution of the so computed values is equal to the previous game and

$$\Pr[S_3] = \Pr[S_2] \ . \tag{4.20}$$

Game$_4$. Now, consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption we have $e^* = e_j$ but $c^* \neq c_j$ for some $j \in [1;q]$. In this game the simulator aborts if c^* and c_j make the hash function collide. We get that

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_h \ . \tag{4.21}$$

Otherwise $\mathcal{B}$ can compute a solution to the SRSA assumption. For $\mathcal{A}$'s forgery $\sigma^* = (r^*, s^*, e^*)$ it now holds that $e^* = e_j$ but $h(c^*) \neq h(c_j)$. We have that

$$\begin{aligned} (s^*)^{e^*} &= uv^{h(c^*)} \bmod n \\ &= \hat{u}^{2\tilde{e}((t_0 - t'_0 h(m^*))\bar{e} + \sum_{i=1}^{q}(h(c_i)-h(c^*))\bar{e}_i} \bmod n \end{aligned}$$

which is equivalent to

$$\left((s^*)\hat{u}^{-2\tilde{e}((t_0-t'_0h(m^*))\bar{e}_j+\sum_{i=1,i\neq j}^{q}(h(c_i)-h(c^*))\bar{e}_{i,j}}\right)^{e_j} \quad = \quad \hat{u}^{2\tilde{e}(h(c_j)-h(c^*))\bar{e}_j} \bmod n.$$

Similar to before we use that $|h(c_j) - h(c^*)| < e_j$ to show that $\gcd(e_j, 2\tilde{e}(h(c_j)-h(c^*))\bar{e}_j)) = 1$. So $\mathcal{B}$ can find $a, b \in \mathbb{Z}$ with $ae_j + b2\tilde{e}(h(c_j) - h(c^*))\bar{e}_j) = 1$ and generate a solution to the SRSA challenge by computing

$$\hat{u}^{1/e_j} = \bar{u}^a \left((s^*)\hat{u}^{-2\tilde{e}(t_0-t'_0h(m^*))\bar{e}_j+\sum_{i=1,i\neq j}^{q}(h(c_i)-h(c^*))\bar{e}_{i,j}}\right)^b .$$

We finally have

$$\Pr[S_4] = \epsilon_{\text{SRSA}}.$$

□

4.3.6 Security Analysis of $\mathcal{S}_{\mathbf{CMB,SDH}}$

Lemma 4.8 *Assume we work in the SDH setting such that the $(t_{SDH}, \epsilon_{SDH})$-SDH assumption holds and $\mathcal{V}$ is a $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining function. Then, the combining signature class as presented in Section 4.2.1 is (q, t, ϵ)-secure against adaptive chosen message attacks provided that*

$$q = q_{SDH}, \quad \epsilon \leq 3\epsilon_{SDH} + 3\epsilon_{comb} + 3q\delta_{comb} + \frac{3q^2}{p}, \quad t = t_{SDH} - \mathsf{T}_{SDH,comb}(q^2).$$

In particular, this means that the SDH based Camenisch-Lysyanskaya scheme is tightly secure against existential forgeries under adaptive chosen message attacks.

Proof. The proof of Lemma 4.8 is the second step in the proof of Theorem 4.3. We consider three types of forgers that after q queries $m_1, \ldots, m_q$ and corresponding responses $(r_1, s_1, t_1), \ldots, (r_q, s_q, t_q)$ partition the set of all possible forgeries $(m^*, (r^*, s^*, t^*))$. $\left(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x\right)$.

Type I Forger $(t^* \notin \{t_1, \ldots, t_q\})$
Suppose $\mathcal{B}$ guesses that $\mathcal{A}$ is a Type I Forger.

Game$_0$. This is the original attack game. By assumption, $\mathcal{A}$ (q, t, ϵ)-breaks $\mathsf{SIG}_{\mathrm{CMB,SDH}}$ when interacting with the signing oracle $\mathcal{O}(SK, \cdot)$. We have that,

$$\Pr[S_0] = \epsilon. \tag{4.22}$$

Game$_1$. Now, $\mathcal{B}$ constructs the values a, b, d using the SDH challenge instead of choosing them randomly. First, $\mathcal{B}$ chooses q random values $t_1, \ldots, t_q \overset{\$}{\leftarrow} \mathbb{Z}_p$ and three random elements $t_0, t'_0, t''_0 \overset{\$}{\leftarrow} \mathbb{Z}_p$. In the following let $\bar{t} := \prod_{k=1}^{q}(t_k + x)$, $\bar{t}_i := \prod_{k=1,k\neq i}^{q}(t_k + x)$ and $\bar{t}_{i,j} := \prod_{k=1,k\neq i,k\neq j}^{q}(t_k + x)$. The simulator computes $a = g_1^{t_0\bar{t}},\ b = g_1^{t'_0\bar{t}},\ c = g_1^{t''_0\bar{t}}$
Since the t_0, t'_0, t''_0 are chosen uniformly at random from $\mathbb{Z}_p$ we have

$$\Pr[S_1] = \Pr[S_0]\ . \tag{4.23}$$

Game$_2$. Now, $\mathcal{B}$ simulates $\mathcal{O}(SK, \cdot)$ by answering $\mathcal{A}$'s signature queries. As before, let $z_j = z(r_j, m_j)$ and $z^* = z(r^*, m^*)$. The simulator $\mathcal{B}$ sets $PK = g_2^x$ and for all $j \in \{1, \ldots, q\}$ it chooses a random $r_j \in \mathcal{R}$ and outputs $\sigma_j = (r_j, s_j, t_j)$ with $s_j = (uv^{r_j}w^{z_j})^{1/(x+t_j)} = g_1^{(t_0+t'_0 r_j+t''_0 z_j)\bar{t}_j}$.

$$\Pr[S_2] = \Pr[S_1]\ . \tag{4.24}$$

Game$_3$. Now, consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, t^*))$. We have that $t^* \notin \{t_1, \ldots, t_q\}$ and

$$e(s^*, PKg_2^{t^*}) = e(ab^{r^*}d^{z(r^*,m^*)}, g_2)$$

which is equivalent to

$$\begin{aligned} s^* &= \left(ab^{r^*}d^{z(r^*,m^*)}\right)^{1/(x+t^*)} \\ &= g_1^{\bar{t}(t_0+t_0'r^*+t_0''z(r^*,m^*))/(x+t^*)}. \end{aligned}$$

$\mathcal{B}$ can now find a solution to the SDH challenge by computing $D \in \mathbb{Z}_p$ with $D \neq 0$ and a polynomial $f'(x) \in \mathbb{Z}_p[x]$ of degree $q-1$ such that $\bar{t} = f'(x)(x+t^*) + D$. We get that

$$g_1^{1/(t^*+x)} = \left((s^*)^{1/(t_0+t_0'r^*+t_0''z(r^*,m^*))} g_1^{-f'(x)}\right)^{1/D}.$$

Finally, we have that

$$\Pr[S_3] = \Pr[S_2] = \epsilon_{\text{SDH}} . \tag{4.25}$$

Plugging in Equations (4.22)–(4.25), we get that $\epsilon = \epsilon_{\text{SDH}}$.

Type II Forger $(t^* = t_i$ and $r^* \neq r_i)$
Now suppose $\mathcal{B}$ expects $\mathcal{A}$ to be a Type II Forger. We only present the differences to the previous proof.

Game₁. First, $\mathcal{B}$ randomly chooses q *distinct* elements $t_1, \dots, t_q \in \mathbb{Z}_p$ and q random elements $r_1, \dots, r_q \in \mathcal{R}$. Additionally, it chooses three random elements $t_0, t_0', t_0'' \in \mathbb{Z}_p$. Next, $\mathcal{B}$ computes $a = g_1^{\left(t_0\bar{t}+\sum_{i=1}^q r_i\bar{t}_i\right)}$, $b = g_1^{\left(t_0'\bar{t}-\sum_{i=1}^q \bar{t}_i\right)}$, and $d = g_1^{t_0''\bar{t}}$ Again,

$$\Pr[S_1] = \Pr[S_0] . \tag{4.26}$$

Game₂. Now $\mathcal{B}$ simulates the signing oracle $\mathcal{O}(SK,\cdot)$. On each signature query m_j with $j \in \{1,\dots,q\}$, $\mathcal{B}$ responds with $\sigma_j = (r_j, s_j, t_j)$ using the precomputed r_j and t_j and computing s_j as

$$s_j = (ab^{r_j}d^{z_j})^{1/(x+t_j)} = g_1^{(t_0+t_0'r_j+t_0''z_j)\bar{t}_j+\sum_{i=1,i\neq j}^q (r_i-r_j)\bar{t}_{i,j}} .$$

$$\Pr[S_2] \geq \Pr[S_1] - q^2/p . \tag{4.27}$$

Game₃. Now consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, t^*))$. By assumption there is a $k \in \{1,\dots,q\}$ with $t^* = t_k$ and $r_k \neq r^*$. Then we have that

$$\begin{aligned} s^* &= \left(ab^{r^*}d^{z^*}\right)^{1/(x+t^*)} = \left(g_1^{(t_0+t_0'r^*+t_0''z^*)\bar{t}+\sum_{i=1}^q (r_i-r^*)\bar{t}_i}\right)^{1/(x+t^*)} \\ &= \left(g_1^{(t_0+t_0'r^*+t_0''z^*)\bar{t}+\sum_{i=1,i\neq k}^q (r_i-r^*)\bar{t}_i+(r_k-r^*)\bar{t}_k}\right)^{1/(x+t_k)} \end{aligned} \tag{4.28}$$

Again we can compute $D \neq 0$ and $f'(x)$ with $\bar{t}_k = f'(x)(x+t_k) + D$ using long division. Therefore,

$$\left(s^* g_1^{-((t_0+t_0'r^*+t_0''z^*)\bar{t}_k+\sum_{i=1,i\neq k}^q (r_i-r^*)\bar{t}_{i,k}+(r_k-r^*)f'(x))}\right)^{1/D(r_k-r^*)} = g_1^{1/(x+t_k)} \tag{4.29}$$

constitutes a solution to the SDH challenge.

$$\Pr[S_3] = \Pr[S_2] = \epsilon_{\text{SDH}} \ . \tag{4.30}$$

Summing up Equations (4.26)–(4.30), we get that $\epsilon \leq \epsilon_{\text{SDH}} + q^2/p$.

Type III Forger $(t^* = t_i \text{ and } r^* = r_i)$
In case $\mathcal{B}$ expects $\mathcal{A}$ to be a Type III Forger, there are only minor differences as compared to the previous proof.

Game₁. First, $\mathcal{B}$ randomly chooses q values $t_1, \ldots, t_q \in \mathbb{Z}_p$ and q random values $z_1, \ldots, z_q \in \mathcal{Z}$. Then, $\mathcal{B}$ draws three random elements t_0, t'_0, t''_0 from $\mathbb{Z}_p$. Next, $\mathcal{B}$ computes a, b, and c as $a = g_1^{t_0\bar{t}+\sum_{i=1}^{q} z_i\bar{t}_i}$, $b = g_1^{t'_0\bar{t}}$, and $c = g_1^{t''_0\bar{t}-\sum_{i=1}^{q}\bar{t}_i}$.

$$\Pr[S_1] = \Pr[S_0] \ . \tag{4.31}$$

Game₂. This game is equal to the previous game except that we require the t_i to be all distinct. We have that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{p} \ . \tag{4.32}$$

Game₃. Now $\mathcal{B}$ simulates the signing oracle. For each queries m_j with $j \in \{1, \ldots, q\}$, $\mathcal{B}$ computes $r_j = z^{-1}(z_j, m_j)$. If $r_j \notin \mathcal{R}$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs the signature $\sigma_j = (r_j, s_j, t_j)$ with s_j being computed as

$$s_j = \left(ab^{r_j}d^{z_j}\right)^{1/(x+t_j)} = g_1^{(t_0+t'_0r_j+t''_0z_j)\bar{t}_j+\sum_{i=1,i\neq j}^{q}(z_i-z_j)\bar{t}_{i,j}} \ .$$

Therefore,

$$\Pr[S_3] \geq \Pr[S_2] - q\delta_{\text{comb}} \ . \tag{4.33}$$

Game₄. Consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, t^*))$. By assumption, there is one index $k \in \{1, \ldots, q\}$ with $t^* = t_k$ and $r^* = r_k$. This game is like the previous one except that $\mathcal{B}$ aborts whenever there occurs a collision $m^* \neq m_k$ such that $z_k = z(r_k, m_k) = z(r_k, m^*) = z^*$. For all t_{comb}-time attackers this happens with probability at most ϵ_{comb}. Therefore,

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\text{comb}} \ . \tag{4.34}$$

It now holds that

$$\left(s^* g_1^{-\left((t_0+t'_0r^*+t''_0z^*)\bar{t}_k+\sum_{j=1,j\neq k}^{q}(z_j-z^*)\bar{t}_{k,j}\right)}\right)^{(x+t_k)} = g_1^{(z_k-z^*)\bar{t}_k}.$$

Compute $D \neq 0$ and $f'(x)$ with $\bar{t}_k = f'(x)(x+t_k) + D$ as before. We now have that

$$\left(s^* g_1^{-\left((t_0+t'_0r^*+t''_0z^*)\bar{t}_k+\sum_{j=1,j\neq k}^{q}(z_j-z^*)\bar{t}_{k,j}+(z_k-z^*)f'(x)\right)}\right)^{1/D(z_k-z^*)} = g_1^{1/(x+t_k)}$$

is a solution to the SDH challenge. Finally,

$$\Pr[S_4] = \epsilon_{\mathrm{SDH}} \ . \tag{4.35}$$

Summing up Equations (4.31)–(4.35), we get that $\epsilon \leq \epsilon_{\mathrm{SDH}} + \epsilon_{\mathrm{comb}} + q\delta_{\mathrm{comb}} + q^2/p$.

Clearly, the running time of $\mathcal{B}$ is $t + \mathsf{T}_{\mathrm{SDH,comb}}(q^2)$ where $\mathsf{T}_{\mathrm{SDH,comb}}(q^2)$ denotes the additional time to compute a, b, d, the signatures in the simulation phase, and the solution to the SDH problem. $\mathsf{T}_{\mathrm{SDH,comb}}(q^2)$ is dominated by q^2 exponentiations in $\mathbb{G}_1$.

□

4.3.7 A Note on Strong Existential Unforgeability

All presented SDH based signatures fulfill our strong notion of security. This is because in the proof we must have $m^* \neq m_i$ only if $t^* = t_i$ and $r^* = r_i$. Now, assume we additionally let $m^* = m_i$. Then, we must also have that $s^* = s_i$. Consequently, $\mathcal{A}$ did not output a valid forgery because $(m^*, \sigma^*) \in \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$. It rather re-sent one of the previous queries together with the corresponding response obtained from the signing oracle.

For the SRSA based signatures, the situation is similar. Consider forgeries $(m^*, (r^*, s^*, e^*))$ where we have for one $i \in [1; q]$ that $e^* = e_i$ and $r^* = r_i$. This is the only case where we must require that $m^* \neq m_i$ to successfully complete the reduction. Now we can argue like before since $m^* = m_i$ of course implies $s^* = s_i$.

4.4 Efficiency Improvements

We now give detailed information on the efficiency improvements that can be achieved by our new reductions.

4.4.1 SRSA-Based Signature Schemes

The best known attack against the SRSA problem is the number field sieve. The general number field sieve describes an attacker that asymptotically can factor n with probability 1 in time $e^{((64/9)^{1/3}+o(1))(l_n)^{1/3}(\log l_n)^{2/3}}$. The special number field sieve has even better asymptotic complexity but can only applied to factor near high powers. In the following we use the recent extrapolation-based approach of ECRYPT II to find appropriate estimates of RSA moduli lengths [49].

The ECRYPT II method assumes that the security level κ is related to the bit-length of the RSA modulus l_n as

$$\kappa = \log_2(e) \cdot (l_n \cdot \ln(2) \cdot 64/9)^{1/3} \cdot (\ln(l_n \cdot \ln(2)))^{2/3} - 14$$

which is equivalent to

$$2^\kappa = e^{(l_n \cdot \ln(2) \cdot 64/9)^{1/3} \cdot (\ln(l_n \cdot \ln(2)))^{2/3}} \cdot 2^{-14}.$$

In the following we assume that the maximal success ratio of any adversary against the SRSA problem is

$$\mathsf{SR}(\mathcal{B}) = \frac{\epsilon_{\mathcal{B}}}{t_{\mathcal{B}}} \leq e^{-(l_n \cdot \ln(2) \cdot 64/9)^{1/3} \cdot (\ln(l_n \cdot \ln(2)))^{2/3}} \cdot 2^{14}. \tag{4.36}$$

In a first step let us recall the running time of the simulator.

- In the original security reduction, the simulator runs in time $t_{\mathcal{A}} + \mathsf{T}_{\text{SRSA,comb}}(q)$ where $\mathsf{T}_{\text{SRSA,comb}}(q)$ is dominated by the time to compute $u^{f(c)} = u^{\prod_{j=1}^{q} e_j + (c_i - c) \prod_{j=1, j \neq i}^{q} e_j}$ for $i \in [1; q]$ given u, $e_1, \ldots, e_q$, and c_i.
- In our new security reduction, the simulator runs in time $t_{\mathcal{A}} + \mathsf{T}_{\text{SRSA,comb}}(q^2)$ where $\mathsf{T}_{\text{SRSA,comb}}(q^2)$ is dominated by the time to compute $u^{f(c)} = u^{\prod_{j=1}^{q} e_j + \sum_{i=1}^{q} (c_i - c) \prod_{j=1, j \neq i}^{q} e_j}$ given u, $e_1, \ldots, e_q$, and $c_1, \ldots, c_q$.

We subsequently assume that $\mathsf{T}_{\text{SRSA,comb}}(q), \mathsf{T}_{\text{SRSA,comb}}(q^2) < t_{\text{SRSA}}/2$. We must clearly have that both $2\mathsf{T}_{\text{SRSA,comb}}(q)$ and $2\mathsf{T}_{\text{SRSA,comb}}(q^2)$ are polynomials in the security parameter. This is because if $\mathsf{T}_{\text{SRSA,comb}}(q), \mathsf{T}_{\text{SRSA,comb}}(q^2) \geq t_{\text{SRSA}}/2$, t_{SRSA} can be computed in polynomial time too and the SRSA problem, contrary to our assumption, can be solved in polynomial time.

4.4.2 Efficiency of the Original Reduction

The original reduction of the SRSA-based combining class states that:

$$t = t_{\text{SRSA}} - \mathsf{T}_{\text{SRSA,comb}}(q), \; \epsilon \leq 9q\epsilon_{\text{SRSA}}/2 + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}.$$

This gives the following bound on $\mathsf{SR}(\mathcal{A})$:

$$\begin{aligned}
\mathsf{SR}(\mathcal{A}) &= \frac{\epsilon}{t} \\
&= \frac{\epsilon}{t_{\text{SRSA}} - \mathsf{T}_{\text{SRSA,comb}}(q)} \\
&\leq \frac{9q\epsilon_{\text{SRSA}}/2 + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}}{t_{\text{SRSA}} - \mathsf{T}_{\text{SRSA,comb}}(q)} \\
&\leq \frac{9q\epsilon_{\text{SRSA}}}{t_{\text{SRSA}}} + \frac{3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}}{t_{\text{SRSA}} - \mathsf{T}_{\text{SRSA,comb}}(q)} \\
&\leq 9q\mathsf{SR}(\mathcal{B}) + \frac{3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}}{t_{\text{SRSA}} - \mathsf{T}_{\text{SRSA,comb}}(q)} \\
&\leq 9q\mathsf{SR}(\mathcal{B}) + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2} \\
&\leq 9q(e^{-(l_n \cdot \ln(2) \cdot 64/9)^{1/3} \cdot (\ln(l_n \cdot \ln(2)))^{2/3}} \cdot 2^{14}) + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}
\end{aligned}$$

We are interested in the minimal choice of l_n, l_r, and l_e. A rough estimate for $|E|$ is given by

$$|E| > \frac{2^{l_e}}{l_e} - \frac{2^{l_e-1}}{l_e-1} = \frac{2^{l_e}(l_e-1) - 2^{l_e-1}l_e}{l_e(l_e-1)} = \frac{2^{l_e-1}(l_e-2)}{l_e(l_e-1)}.$$

For $3 < l_e$ we get $\frac{1}{2} < \frac{l_e-2}{l_e-1}$ and

$$|E| > \frac{2^{l_e-2}}{l_e}.$$

With $2^{l_n-1} \leq n < 2^{l_n}$ and $2^{l_q-1} \leq q < 2^{l_q}$ we get that

$$\begin{aligned}
& 9q(e^{-(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}} \cdot 2^{14}) + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{|E|} + 9\cdot 2^{2-l_n/2} \\
\leq\ & 2^{l_q+4}(e^{-(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}} \cdot 2^{14}) + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{2^{2l_q+2}}{|E|} + 2^{6-l_n/2} \\
\leq\ & 2^{l_q+4}\cdot 2^{-\log_2(e)\cdot(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}+14} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{2^{2l_q+2}}{|E|} + 2^{6-l_n/2} \\
\leq\ & 2^{l_q+18-\log_2(e)\cdot(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{2^{2l_q+2}l_e}{2^{l_e-2}} + 2^{6-l_n/2} \\
\leq\ & 2^{l_q+18-\log_2(e)\cdot(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + 2^{2l_q+4-l_e}l_e + 2^{6-l_n/2}.
\end{aligned}$$

This value should be smaller than $2^{-\kappa}$. Now assume we use the concrete combining functions EX2 and EX3. In both cases we get that $\epsilon_{\text{comb}} = 0$. For EX2 we also get that $\delta_{\text{comb}} = 0$, whereas for EX3 we have that $3\delta_{\text{comb}} < 2^{l_m-l_r+2}$.

To have each of the summands be smaller than $2^{-\kappa-2}$ (resulting in a sum smaller than $2^{-\kappa}$) we obviously need

$$l_e - \log_2(l_e) > 2l_q + 6 + \kappa$$

and at least

$$16 + 2\kappa < l_n.$$

For EX3 (EX2 already specifies $l_r = l_m$) we additionally get that

$$l_r > \kappa + l_m + 4.$$

The first term gives us

$$l_q + 20 + \kappa < \log_2(e)\cdot(l_n\cdot\ln(2)\cdot 64/9)^{1/3}\cdot(\ln(l_n\cdot\ln(2)))^{2/3}$$

what results in $((l_q+\kappa+20)/\log_2(e))^3 < (l_n\cdot\ln(2)\cdot 64/9)\cdot(\ln(l_n\cdot\ln(2)))^2$ and

$$\frac{9\cdot(l_q+\kappa+20)^3}{64\cdot\ln(2)\cdot(\log_2(e))^3} < l_n\cdot(\log l_n)^2$$

to bound $\mathsf{SR}(\mathcal{A}) \leq 2^{-\kappa}$. In Table 4.3 we provide exact values for l_r, l_e, and l_n for $q = 2^{30}$, $q = 2^{40}$, and $q = 2^{50}$.

	Security level $\kappa = 80$						Security level $\kappa = 128$					
	Orig. Reduction			New Reduction			Orig. Reduction			New Reduction		
q	l_r	l_e	l_n	l_r	l_e	l_n	l_r	l_e	l_n	l_r	l_e	l_n
2^{30}	244	154	2593	274	154	1395	292	202	5673	322	202	3652
2^{40}	244	174	3095	284	174	1395	292	222	6470	332	222	3652
2^{50}	244	194	3652	294	194	1395	292	242	7331	342	242	3652

Table 4.3: Efficiency improvements for the Camenisch-Lysyanskaya signature scheme (EX2) using our new security reduction. The table presents the bit sizes of the modulus n and the random values r and e for different numbers q of signing queries and a target security level of $2^{-\kappa}$. The given values also hold for the Fischlin scheme (EX1) (except that by design $l_r = l_m$).

4.4.3 Efficiency of the New Reduction

The new reduction of the SRSA-based combining class has the following properties:

$$t \approx t_{\mathrm{SRSA}} - \mathsf{T}_{\mathrm{SRSA,comb}}(q^2),\ \epsilon \leq 9\epsilon_{\mathrm{SRSA}}/2 + 3\epsilon_{\mathrm{comb}} + 3q\delta_{\mathrm{comb}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}.$$

When using EX3, our new reduction affects the first and the second summand. We get

$$\frac{9 \cdot (\kappa + 20)^3}{64 \cdot \ln(2) \cdot (\log_2(e))^3} < l_n \cdot (\log l_n)^2$$

and

$$l_r > \kappa + l_m + l_q + 4.$$

The last equation shows that the required size of l_r is significantly worse than in the original reduction (we have to account for the number of signature queries q) for EX3. However for EX1 we have that $\delta_{\mathrm{comb}} = 0$ and it always holds that $l_r = l_m$. So we can fully benefit from the improvements resulting from $9 \cdot (\kappa + 20)^3/(64 \cdot \ln(2) \cdot (\log_2(e))^3) < l_n \cdot (\log l_n)^2$.

4.4.4 SDH-Based Signature Schemes

The best known attack on SDH is Cheon's attack that describes an attacker $\mathcal{P}$ such that

$$\mathsf{SR}(\mathcal{P}) = \frac{\epsilon_{\mathcal{P}}}{t_{\mathcal{P}}} = \Omega(\sqrt{\frac{q}{p}}). \tag{4.37}$$

Based on this result we make the assumption that $\sqrt{\frac{q}{p}}$ is the maximal success ratio of any adversary against the SDH problem

$$\mathsf{SR}(\mathcal{B}) \leq \sqrt{\frac{q}{p}}. \tag{4.38}$$

Let us again start by bounding the running time of the simulator.

- In the original security reduction, the simulator runs in time $t_{\mathcal{A}} + \mathsf{T}_{\text{SDH,comb}}(q)$ where $\mathsf{T}_{\text{SDH,comb}}(q)$ is dominated by the time to compute $g^{\prod_{j=1}^{q}(x-t_j)+(c_i-c)\prod_{j=1,j\neq i}^{q}(x-t_j)}$ for $i \in [1;q]$ given $g, g^2, \ldots, g^q$, $t_1, \ldots, t_q$, and c_i.

- In our new security reduction, the simulator runs in time $t_{\mathcal{A}} + \mathsf{T}_{\text{SDH,comb}}(q^2)$ where $\mathsf{T}_{\text{SDH,comb}}(q^2)$ is dominated by the time to compute $g^{\prod_{j=1}^{q}(x-t_j)+\sum_{i=1}^{q}(c_i-c)\prod_{j=1,j\neq i}^{q}(x-t_j)}$ given $g, g^2, \ldots, g^q$, $t_1, \ldots, t_q$, and $c_1, \ldots, c_q$.

Similar to before, we subsequently assume that $\mathsf{T}_{\text{SDH,comb}}(q), \mathsf{T}_{\text{SDH,comb}}(q^2) < t_{\text{SDH}}/2$.

4.4.5 Efficiency of the Original Reduction

The original reduction of the SDH-based combining class states that:

$$q = q_{\text{SDH}},\ t = t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q),\ \epsilon \leq 3q\epsilon_{\text{SDH}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p}. \tag{4.39}$$

This gives rise to the following bound on $\mathsf{SR}(\mathcal{A})$:

$$\begin{aligned}
\mathsf{SR}(\mathcal{A}) &= \frac{\epsilon}{t} \\
&= \frac{\epsilon}{t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q)} \\
&\leq \frac{3q\epsilon_{\text{SDH}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p}}{t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q)} \\
&\leq \frac{6q\epsilon_{\text{SDH}}}{t_{\text{SDH}}} + \frac{3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p}}{t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q)} \\
&\leq 6q\mathsf{SR}(\mathcal{B}) + \frac{3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p}}{t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q)} \\
&\leq 6q\mathsf{SR}(\mathcal{B}) + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p} \\
&\leq 6q\sqrt{\frac{q}{p}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p}
\end{aligned}$$

We are interested in the minimal choice of p. With $2^{l_p-1} \leq p < 2^{l_p}$ and $2^{l_q-1} \leq q < 2^{l_q}$

	$\kappa = 80$		$\kappa = 128$	
	Orig. Red.	New Red.	Orig. Red.	New Red.
q	l_p	l_p	l_p	l_p
2^{30}	258	198	354	294
2^{40}	288	208	384	304
2^{50}	318	218	414	314

Table 4.4: Efficiency improvements for the SDH-based Camenisch-Lysyanskaya signature scheme. l_p is the bit size of the group order p for different numbers q of signing queries and a target security level of $2^{-\kappa}$.

we get that

$$\begin{aligned}
6q\sqrt{\frac{q}{p}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p} &\leq 6q\sqrt{\frac{q}{p}} + \frac{3q^2}{p} \\
&\leq 2^{l_q+3}\sqrt{\frac{2^{l_q}}{2^{l_p-1}}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3 \cdot 2^{2l_q}}{2^{l_p-1}} \\
&\leq 2^{l_q+3}\sqrt{\frac{2^{l_q}}{2^{l_p-1}}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{2^{2l_q+2}}{2^{l_p-1}} \\
&\leq 2^{l_q+3+l_q/2-(l_p-1)/2} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + 2^{2l_q+2-l_p+1} \\
&\leq 2^{3l_q/2+7/2-l_p/2} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + 2^{2l_q+3-l_p}.
\end{aligned}$$

This value should be smaller than $2^{-\kappa}$. EX1 gives $l_r = l_m$ and $\epsilon_{\text{comb}} = \delta_{\text{comb}} = 0$. For $l_p \geq 3l_q + 7 + 2\kappa$ we get $2^{3l_q/2+7/2-l_p/2} \leq 2^{-\kappa}$ and $2^{2l_q+3-l_p} \leq 2^{2l_q+3-(3l_q+7+2\kappa)} = 2^{-4-l_q-2\kappa} \leq 2^{-\kappa}$. Thus

$$l_p \geq 3l_q + 8 + 2\kappa$$

suffices to bound $\mathsf{SR}(\mathcal{A}) \leq 2^{-\kappa}$ when using EX1. We give exact values for l_p in Table 4.4.

4.4.6 Efficiency of the New Reduction

The new reduction of the SDH-based combining class states:

$$q = q_{\text{SDH}},\ t = t_{\text{SDH}} - \mathsf{T}_{\text{SDH,comb}}(q^2),\ \epsilon \leq 3\epsilon_{\text{SDH}} + 3\epsilon_{\text{comb}} + 3q\delta_{\text{comb}} + \frac{3q^2}{p}.$$

In a similar way as before we get

$$\begin{aligned}
6\sqrt{\frac{q}{p}} + 3\epsilon_{\text{comb}} + 3\delta_{\text{comb}} + \frac{3q^2}{p} &\leq 6\sqrt{\frac{q}{p}} + \frac{3q^2}{p} \\
&\leq 6\sqrt{\frac{2^{l_q}}{2^{l_p-1}}} + \frac{3\cdot 2^{2l_q}}{2^{l_p-1}} \\
&\leq 6\sqrt{\frac{2^{l_q}}{2^{l_p-1}}} + \frac{2^{2l_q+2}}{2^{l_p-1}} \\
&\leq 2^{l_q/2+3-(l_p-1)/2} + 2^{2l_q+2-l_p+1} \\
&\leq 2^{l_q/2+7/2-l_p/2} + 2^{2l_q+3-l_p}.
\end{aligned}$$

Now we can set $l_p \geq l_q+7+2\kappa$ and get that $2^{l_q/2+7/2-l_p/2} \leq 2^{-\kappa}$ and $2^{2l_q+3-l_p} \leq 2^{2l_q+3-(l_q+7+2\kappa)} = 2^{-4+l_q-2\kappa}$. If we assume that

$$\kappa \geq l_q - 4$$

which is reasonable and typically given in concrete security analyses ($\kappa = 80$ and $l_q \leq 50 \leq 80$) we get $2^{-4+l_q-2\kappa} \leq 2^{-\kappa}$. Thus

$$l_p \geq l_q + 8 + 2\kappa$$

suffices to bound $\mathsf{SR}(\mathcal{A}) \leq 2^{-\kappa}$ for EX1. We give exact values in Table 4.4.

Chapter 5

Two-Tier Signature Schemes

The design of new signature schemes in the standard model is a seemingly difficult task. What has proven very useful in the past is the application of transformations from other cryptographic primitives to signature schemes.

Two Important Signature Transformations. Let us consider two important transformations that turn signature schemes with weaker security properties to schemes with strong security features. The first example is the generic transformation by Even *et al.* (subsequently denoted as gCMA→aCMA transformation) which constructs EMUF-aCMA-secure signature schemes from EMUF-gCMA-secure schemes. The main idea is to use the weakly secure signature scheme to sign a fresh one-time public key *opk*. The corresponding secret key *osk* is then used to sign the message m. The final signature σ consists of the signature σ_1 on *opk*, the one-time signature σ_2 on m, and *opk*: $\sigma = (\sigma_1, \sigma_2, opk)$.

Another general transformation (in the following called aCMA→STR-aCMA transformation) was presented by Huang *et al.* [67] and independently by Bellare and Shoup [13]. It generate a strongly EMUF-aCMA-secure signature scheme – where adversaries are also allowed to output new signatures on previously queried messages – from a EMUF-aCMA-secure signature scheme and a strongly secure one-time signature scheme. Similar to the Even *et al.* transformation the construction proceeds in two steps. First, the EMUF-aCMA-secure signature scheme is used to generate a signature σ_1 on $m||opk$ where m is the message to be signed and *opk* is a fresh one-time public key. Next, the signer generates a one-time signature σ_2 on σ_1 using *osk*. Again the final signature σ consist of three elements: $\sigma = (\sigma_1, \sigma_2, opk)$.

We note that both of the above transformations essentially rely on one-time signature schemes. Since signature schemes trivially imply one-time signature schemes one could at least in the Even *et al.* transformation utilize a new instance of the basic signature scheme as a substitute for a dedicated one-time signature. However, since they must only provide much weaker security guarantees than classical signature schemes, one time signature schemes are in general more efficient than usual signature schemes [87, 89].

Drawbacks of the Existing Transformations. With respect to signature size, both of the above transformations have two serious drawbacks. First, the final signature must always contain the one-time public key to allow for a verification of σ_1 and σ_2. Second, the security reduction looses a factor of q (where q is the number of signature queries made by the attacker) which in turn transfers to larger parameter sizes for equal security guarantees as compared to tight reductions. To illustrate the latter problem consider how a simulator would embed the challenge for the one-time signature. In the attack game this situation occurs whenever the adversary re-uses one of the opk_i (as part of signature $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, opk_i)$) with $i \in \{1, \ldots, q\}$ in his forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, opk^*)$ such that $opk^* = opk_i$ and $\sigma_1^* = \sigma_{1,i}$. In this case, security cannot be reduced to the underlying signature scheme since σ_1^* is not a signature on a new message. The simulator's strategy to solve this problem is to guarantee that such a re-use of $\sigma_{1,i}$ always corresponds to a break of the one-time signature scheme. Indeed, since $\sigma_1^* = \sigma_{1,i}$ and $opk^* = opk_i$ we must have by assumption that $\sigma^* \neq \sigma_i$ (otherwise σ^* would not be a forgery) and $\sigma_{2,i}$ must be a valid signature on a new message what contradicts the security guarantees of the one-time signature scheme. Now to embed the one-time signature challenge, the simulator at first guesses i (indicating which value s_i the attacker is going to re-use in the forgery) and uses the one-time oracle to obtain opk_i and to generate $\sigma_{2,i}$. As a consequence of this guess, the simulator only succeeds with probability $1/q$ in the security reductions.

Two-tier Signature Schemes. Bellare and Shoup [13] showed how to tackle the latter problem using a new primitive called two-tier signature scheme.[1] Two-tier signature schemes can be regarded as an intermediate primitive located between one-time signature schemes and usual signature schemes. The signer first generates a long term public key pair (SK, PK). Whenever he desires he can construct fresh key material for a one-time signature scheme (osk_i, opk_i) and sign a message m_i. The distinguishing property of two-tier signature schemes (as compared to usual one-time signature schemes) is that the one-time keys are *not pairwisely independent* given (SK, PK). As a consequence the so generated one-time signatures share the following useful property: if any of the issued one-time signature schemes (identified by its key material) is broken, so is the two-tier signature scheme. As two-tier signature schemes can be build from classical three move identification protocols there exist efficient variants secure under standard assumptions like the discrete logarithm assumption or the factoring assumption.

Contribution. In this chapter, we present new instantiations of two-tier signature schemes from chameleon hash functions. Using our construction, we for the first time show how to solve the above problems *simultaneously*. First, we present a new security notion for two-tier signature schemes called static security (formally defined in Section 5.1) that is much weaker than the original definition (adaptive security as defined in Section 2.12.1). Then we show how to build statically secure two-tier signature schemes from chameleon hash functions. When we apply our construction to the Even *et al.* transformation (as a substitute for classical one-time signatures) we directly obtain the Shamir-Tauman transformation. This result for the first time fully clarifies the relationship between the Even *et al.* transformation

[1]The original motivation for two-tier signature schemes was the Fiat-Shamir heuristic [53]. Bellare and Shoup analyzed the security of signature schemes that are generated via the Fiat-Shamir heuristic while *not* assuming random oracles. Instead the hash function in the Fiat-Shamir heuristic is solely required to be collision-resistant.

and the Shamir-Tauman transformation. A previous analysis by Catalano *et al.* [35] already observed that chameleon hash functions give rise to one-time signature schemes but it did not explain why the security Shamir-Tauman reduction tightly reduces to the underlying security assumption (as compared to the Even *et al.* construction that looses a factor of q in the security proof). Next, we present a general transformation from statically secure two-tier signature schemes to adaptively secure schemes. We so obtain efficient and adaptively secure two-tier signature schemes from standard assumptions via a new construction.

In contrast to the existing construction that is derived from the Fiat-Shamir heuristic, both of our chameleon based constructions (the statically secure and the adaptively secure) have a very useful property: given the message m and the corresponding one-time signature on m one can easily *compute* the public one-time key. If we use this construction as a stand-alone substitute for one-time signatures the advantage is small. For verification we still must transmit the corresponding one-time public key: the verifier uses message and signature to compute his version of the one-time public key and compares it with the one received. However, in cascaded constructions like the signature transformations by Even *et al.* and Bellare-Shoup/Huang *et al.* our construction pays off.

To (informally) explain this, recall that after applying one of these transformations a signature consists of $\sigma = (\sigma_1, \sigma_2, opk)$ where σ_1 is a classical signature on message opk (respectively $m||opk$) and σ_2 is a one-time signature on message m (respectively σ_1) that can be verified with opk. The security properties of the classical signature scheme guarantee that σ_1 only correctly verifies under public key PK if opk (respectively $m||opk$) is exactly the message that was used to produce σ_1. Any other message will result in a verification fail. In our chameleon hash based construction we can, given m (respectively σ_1) and σ_2, compute a candidate for opk. Whether our candidate actually is the correct one-time public key is implicitly checked when verifying σ_1. Therefore we do not need to transmit opk! Also it is well known that chameleon hash functions can be generated from standard assumptions like the discrete logarithm assumption or the factoring assumption which are weaker than most of the existing assumptions used to construct signature schemes. Therefore two-tier signature schemes derived from chameleon hash functions can be based on the same well-studied standard assumptions as those derived from three move identification protocols. Moreover, the most efficient instantiations proposed in [13] rely on 'one-more' assumptions like the one-more discrete logarithm assumption or the one-more root inversion assumption. In contrast, we can easily provide an instantiation example that relies on the mere discrete logarithm assumption using existing chameleon hash functions. It is not only based on weaker assumptions but, exploiting the special properties of two-tier signature schemes based on chameleon hash functions, *also* more efficient when applied in cascaded constructions (like the $\mathsf{aCMA}\to\mathsf{STR\text{-}aCMA}$ transformation) than the existing solutions. We also stress, that, as we do not have to derive security of the two-tier signature scheme from properties of an underlying identification protocol, our security analysis is somewhat more simple. In particular, we do not have to restrict ourself to protocols that are secure under concurrent attacks. Our security guarantees directly follow from the security of the chameleon hash functions. Similar to the Bellare-Shoup construction, the cost for our approach is that SK and PK

have to be added to the long-term key material. We stress that efficiency improvements for transformations generally have great impact. In contrast to improvements made to a single, concrete scheme enhanced transformations automatically improve *all* signature schemes that rely on this transformation.

RELATED WORK. Our result clarifies and improves several previous results. Besides improving the Bellare-Shoup transformation, it extends the result of Catalano *et al.* from 2008 who showed how to obtain *usual* one-time signatures from chameleon hash functions [35]. Our transformation from statically secure two-tier signature schemes to adaptively secure schemes can be regarded as a generalization of the Even *et al.* transformation (1989) that only works for classical weakly secure (one-time) signature schemes [51]. Furthermore our final transformation from fully to strongly secure signature schemes can be regarded as a Shamir-Tauman-like (2001) [96] improvement of the Huang *et al.* construction from 2008. We use chameleon hash functions to shorten the signature size *and* tighten the security proof.

5.1 Statically Secure Two-Tier Signature Schemes - A New Weak Security Notion

We now introduce our new security notion. The main difference to the Bellare-Shoup definiton is that the adversary may not query one-time public key prior to signatures. One-time public keys are only output together with the signatures on the queried messages.

STATICALLY SECURE TWO-TIER SIGNATURE SCHEMES. Our new security definition is defined by the following attack game.

Public Key and Signature Generation. In the first phase the adversary is given a public key PK.

Signature queries. Next, the adversary can adaptively access the oracle $\mathcal{O}_{SK,\text{static}}(\cdot)$ q times. Given a message m_i with $i \in [1;q]$, $\mathcal{O}_{SK,\text{static}}(\cdot)$ outputs a fresh one-time key opk_i and a one-time signature σ_i on m_i such that $\mathsf{TT.Verify}(PK, opk_i, m_i, \sigma_i) = 1$.

Output. The attacker outputs a forgery (m^*, σ^*) such that there exists $i \in [1;q]$ with $m^* \neq m_i$ and $\mathsf{TT.Verify}(PK, opk_i, m^*, \sigma^*) = 1$.

We denote the success probability of an adversary $\mathcal{A}$ (taken over the random coins of the challenger and the adversary) to win the static security game as $Adv_{\mathsf{TT},\mathcal{A},\text{static}}$ and the adaptive security game as $Adv_{\mathsf{TT},\mathcal{A},\text{adapt}}$.

Definition 5.1 (Secure Two-Tier Signature Scheme) *An adversary $\mathcal{A}$ is said to (q,t,ϵ)-break the static security of a given two-tier signature scheme TTif $\mathcal{A}$ has success probability $Adv_{\mathsf{TT},\mathcal{A},static} = \epsilon$ after generating at most q one-time queries and running in time t. $\mathsf{TT}$is said to be (q,t,ϵ) statically secure if there exists no PPT adversary that (q,t,ϵ)-breaks the static security of TT. We call a two-tier signature scheme strongly secure if the generated one-time signature schemes are strongly secure (i.e. $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$).*

5.2 Construction of Statically Secure Two-Tier Signature Schemes from Chameleon Hash Functions

We now show how to generate statically secure two-tier signature schemes from chameleon hash functions. The idea behind our construction is the following. Whenever the signer wants to generate new one-time key material it evaluates the chameleon hash function on random input values (i.e. message m' and randomness r'). The result of this computation is the one-time public key. The corresponding input values form the one-time secret key. To compute a one-time signature on message m we use the secret key of the chameleon hash function to compute a collision r such that (m, r) and (m', r') map to the same output value opk. More formally: let $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ be a chameleon hash function. Then the following construction yields a two-tier signature scheme $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$.

- $\mathsf{TT.KeyGen}(1^\kappa)$: computes $\mathsf{CH.Gen}(1^\kappa) = (SK_{\mathsf{CH}}, PK_{\mathsf{CH}})$ and sets $SK = SK_{\mathsf{CH}}$ and $PK = PK_{\mathsf{CH}}$.
- $\mathsf{TT.OTKeyGen}(SK, PK)$: randomly chooses message $m' \in \mathcal{M}$ and randomness $r' \in \mathcal{R}$ and computes $opk = \mathsf{CH.Eval}(PK, m', r')$. The secret one-time key is set to $osk = (m', r')$.
- $\mathsf{TT.Sign}(SK, osk, m)$: parses osk as (m', r'). Then it computes a collision for opk using $\mathsf{CH.Coll}$ as $\sigma = \mathsf{CH.Coll}(SK, r', m', m) = r$.
- $\mathsf{TT.Verify}(PK, opk, m, \sigma)$: checks if $\mathsf{CH.Eval}(PK, m, \sigma)$ equals opk. In this case it outputs 1, otherwise 0.

Theorem 5.2 *Let* $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ *be a* $(t_{\mathsf{CH}}, \epsilon_{\mathsf{CH}})$*-secure chameleon hash function. Then for any polynomial* $q = q(\kappa)$ *the above construction yields a* (q, t, ϵ) *statically secure two-tier signature scheme* TT *with* $t = t_{\mathsf{CH}}$ *and* $\epsilon = \epsilon_{\mathsf{CH}}$*. Moreover, if the chameleon hash function is strongly secure so is* TT*.*

Proof. Assume there exists a successful adversary $\mathcal{A}$ against TT. We show that we can build a simulator $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of the chameleon hash function.

By assumption, $\mathcal{B}$ is given the public key PK_{CH} of the chameleon hash function. In the first step $\mathcal{B}$ gives $PK = PK_{\mathsf{CH}}$ to $\mathcal{A}$. Next $\mathcal{B}$ simulates oracle $\mathcal{O}_{SK,\text{static}}(\cdot)$: whenever $\mathcal{A}$ queries message $m_j \in \mathcal{M}$ with $j \in [1; n]$, $\mathcal{B}$ chooses a random $\sigma_j \in \mathcal{R}$ and computes $opk_j = \mathsf{CH.Eval}(PK, m_j, \sigma_j)$. The one-time public key opk_j and the one-time signature σ_j on m_j are given to $\mathcal{A}$. Observe that these values are exactly distributed as in the original attack game. Finally, $\mathcal{A}$ outputs a forgery (m^*, σ^*) such that there exists $i \in [1; q]$ with $\mathsf{TT.Verify}(PK, opk_i, m^*, \sigma^*) = 1$. But this means that $\mathsf{CH.Eval}(PK, m_i, \sigma_i) = opk_i = \mathsf{CH.Eval}(PK, m^*, \sigma^*)$ and since we have $m^* \neq m_i$ (or $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$ in the case of strong security), $\mathcal{B}$ has found a collision of the chameleon hash function. This concludes the proof of Theorem 5.2. $\square$

5.3 Generalizing the Shamir-Tauman Transformation: A New Transformation from Weakly Secure Signature Schemes to Fully Secure Schemes using Statically Secure Two-tier Signature Schemes

We now show how to generically substitute the weak one-time signature scheme in the $\mathsf{gCMA}\rightarrow\mathsf{aCMA}$ transformation with statically secure two-tier signatures. This tightens the security reduction. Let $\mathsf{SIG}_{\text{weak}} = (\mathsf{SIG.KeyGen}_{\text{weak}}, \mathsf{SIG.Sign}_{\text{weak}}, \mathsf{SIG.Verify}_{\text{weak}})$ be a weakly secure signature scheme. Assume $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$ is a statically secure two-tier signature scheme. Then we can build the following fully secure signature scheme $\mathsf{SIG} = (\mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$.

- $\mathsf{SIG.KeyGen}(1^\kappa)$: run $\mathsf{SIG.KeyGen}_{\text{weak}}(1^\kappa)$ and obtain $(PK_{\text{weak}}, SK_{\text{weak}})$. Next, run $\mathsf{TT.KeyGen}(1^\kappa)$ and obtain SK_{TT} and PK_{TT}. Set $SK = (SK_{\text{weak}}, SK_{\mathsf{TT}})$ and $PK = (PK_{\text{weak}}, PK_{\mathsf{TT}})$.

- $\mathsf{SIG.Sign}(SK, m)$: to sign a message m, run $\mathsf{TT.OTKeyGen}(SK_{\mathsf{TT}}, PK_{\mathsf{TT}})$ to get a new key pair (osk, opk). Compute the signatures $\sigma_1 = \mathsf{SIG.Sign}(SK_{\text{weak}}, opk)$[2] and $\sigma_2 = \mathsf{TT.Sign}(SK_{\mathsf{TT}}, osk, m)$. Output the final signature as $\sigma = (\sigma_1, \sigma_2, opk)$.

- $\mathsf{SIG.Verify}(PK, m, \sigma)$: if it holds that $\mathsf{SIG.Verify}_{\text{weak}}(PK_{\text{weak}}, opk, \sigma_1) = 1$ and at the same time $\mathsf{TT.Verify}(PK_{\mathsf{TT}}, opk, m, \sigma_2) = 1$ output 1. Otherwise output 0.

Theorem 5.3 *Suppose that* $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$ *is a two-tier signature scheme that is* $(t', \epsilon_{\mathsf{TT}})$ *statically secure. Furthermore, assume that* $\mathsf{SIG}_{weak} = (\mathsf{SIG.KeyGen}_{weak}, \mathsf{SIG.Sign}_{weak}, \mathsf{SIG.Verify}_{weak})$ *is a signature scheme that is* $(q_{weak}, t', \epsilon_{weak})$ *weakly secure. Then the above construction yields a* (q, t, ϵ) *fully secure signature scheme* $\mathsf{SIG} = (\mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ *provided that* $q = q_{weak}$, $t \approx t'$, $\epsilon \leq 2\epsilon_{weak} + 2\epsilon_{\mathsf{TT}}$. *The security reduction is tight.*

The proof of Theorem 5.3 is very similar to the proofs in [51, 96].

Proof. Let $\mathcal{A}$ be an attacker against SIG. Suppose $\mathcal{A}$ queries messages $m_1, \ldots, m_q$. Moreover let $\sigma_1 = (\sigma_{1,1}, \sigma_{2,1}, opk_1), \ldots, \sigma_q = (\sigma_{1,q}, \sigma_{2,q}, opk_q)$ be the corresponding signatures on these messages. The simulator $\mathcal{B}$ at first draws uniformly at random $b \in \{1, 2\}$ indicating its guess of $\mathcal{A}$'s final forgery $(m^*, \sigma^* = (\sigma_1^*, \sigma_2^*, opk^*))$. With probability $\geq 1/2$ this guess is correct.

If $b = 1$, $\mathcal{B}$ assumes that $opk^* \notin \{opk_1, \ldots, opk_q\}$. In this case $\mathcal{B}$ can use $\mathcal{A}$ to break the security of the weakly secure signature scheme $\mathsf{SIG}_{\text{weak}}$. In the first step $\mathcal{A}$ generates SK_{TT} and PK_{TT} using the algorithm $\mathsf{TT.KeyGen}(1^\kappa)$. Then it computes q key pairs

[2]As [51] point out this value is message independent and can be precomputed yielding offline/online signatures. In this case, the only remaining task to complete the signature generation is to sign m using a one-time signature scheme what greatly shortens response times.

$(osk_1, opk_1), \ldots, (osk_q, opk_q)$ using $\mathsf{TT.OTKeyGen}(SK_{\mathsf{TT}}, PK_{\mathsf{TT}})$. In the next step $\mathcal{B}$ sends $opk_1, \ldots, opk_q$ to $\mathcal{O}_{\text{weak}}(\cdot)$ and receives the public key $\overline{PK}$ and signatures $s_1, \ldots, s_q$. $\mathcal{B}$ sets $\sigma_{1,i} = s_i$ for all $i \in [1;q]$ and outputs $PK_{\text{weak}} = \overline{PK}$. Now when $\mathcal{A}$ queries message m_i, $\mathcal{B}$ simply computes $\sigma_{2,i} = \mathsf{TT.Sign}(SK_{\mathsf{TT}}, osk_i, m_i)$ and outputs $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, opk_i)$. Observe that all values are distributed exactly like in the original attack game. Eventually $\mathcal{A}$ outputs $(m^*, \sigma^* = (\sigma_1^*, \sigma_2^*, opk^*))$. Since by assumption $opk^* \notin \{opk_1, \ldots, opk_q\}$, (opk^*, σ_1^*) breaks the security of $\mathsf{SIG}_{\text{weak}}$.

If $b = 2$, $\mathcal{B}$ assumes that there exists an index $j \in [1;q]$ such that $opk^* = opk_j$. In this case $\mathcal{B}$ can use $\mathcal{A}$ to break the security of the statically secure two-tier signature scheme TT. In the first step $\mathcal{A}$ generates SK_{weak} and PK_{weak} using $\mathsf{SIG.KeyGen}(1^\kappa)$. Let $\overline{PK}$ be the challenge key for the two-tier signature scheme. $\mathcal{B}$ sets $PK_{\mathsf{TT}} = \overline{PK}$. Now when $\mathcal{A}$ queries message m_i, $\mathcal{A}$ sends m_i to oracle $\mathcal{O}_{SK,\text{static}}(\cdot)$ and receives opk_i and s_i. $\mathcal{B}$ sets $\sigma_{2,i} = s_i$. Then $\mathcal{B}$ uses SK_{weak} to compute $\sigma_{1,i} = \mathsf{SIG.Sign}(SK_{\text{weak}}, opk_i)$ before outputting $\sigma = (\sigma_{1,i}, \sigma_{2,i}, opk_i)$. Observe that all values are distributed exactly as in the original attack game. Eventually $\mathcal{A}$ outputs $(m^*, \sigma^* = (\sigma_1^*, \sigma_2^*, opk^*))$. Since by assumption $opk^* = opk_i$ for some $i \in [1;q]$, (m^*, σ_2^*) breaks the security of TT.

This concludes the proof of Theorem 5.3. □

Corollary 5.4 *If we apply our construction of statically secure two tier signature schemes based on chameleon hash functions to the above transformation and exploit the special properties of the construction (the one-time public key can be generated from the one-time signature and the message) we immediately get the Shamir-Tauman construction.*

In Section 5.6.1 we will provide a detailed security proof of a new transformation for constructing strongly secure signature schemes. In the proof, we in detail show how to exploit the properties of chameleon hash based two-tier signatures. This technique can easily be adapted to the conceptually more simple proof of Corollary 5.4.

5.4 A New Transformation from Statically to Adaptively Secure Two-Tier Signature Schemes

We now show how to obtain adaptively secure two-tier signature schemes from chameleon hash functions. To this end we present a transformation that constructs adaptively secure two-tier signature schemes from statically secure schemes. The main idea is to use a two key technique in the spirit of the $\mathsf{gCMA} \longrightarrow \mathsf{aCMA}$ transformation. As a consequence we need to double the size of the long-term key material.

5.4.1 Transformation from Statically Secure to Adaptively Secure Two-Tier Signature Schemes

Let $\mathsf{TT}_{\text{static}} = (\mathsf{TT.KeyGen}_{\text{static}}, \mathsf{TT.OTKeyGen}_{\text{static}}, \mathsf{TT.Sign}_{\text{static}}, \mathsf{TT.Verify}_{\text{static}})$ be a statically secure two-tier signature scheme. Then we can build the following adaptively secure two-tier signature scheme $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$.

- $\mathsf{TT.KeyGen}(1^\kappa)$: runs $\mathsf{TT.KeyGen}_{\text{static}}(1^\kappa)$ twice and obtains (SK_1, PK_1) and (SK_2, PK_2). It sets $SK = (SK_1, SK_2)$ and publishes $PK = (PK_1, PK_2)$.
- $\mathsf{TT.OTKeyGen}(SK, PK)$: given $SK = (SK_1, SK_2)$ and $PK = (PK_1, PK_2)$ this algorithm computes $(osk, opk) \leftarrow \mathsf{TT.OTKeyGen}_{\text{static}}(SK_1, PK_1)$.
- $\mathsf{TT.Sign}(SK, osk, m)$: given $SK = (SK_1, SK_2)$ and osk this algorithm first computes a new one-time key pair $(osk', opk') \leftarrow \mathsf{TT.OTKeyGen}_{\text{static}}(SK_2, PK_2)$. Then it computes the signature $\sigma_1 = \mathsf{TT.Sign}_{\text{static}}(SK_1, osk, opk')$ and $\sigma_2 = \mathsf{TT.Sign}_{\text{static}}(SK_2, osk', m)$ and outputs $\sigma = (\sigma_1, \sigma_2, opk')$.
- $\mathsf{TT.Verify}(PK, opk, m, \sigma)$: if it holds that $\mathsf{TT.Verify}_{\text{static}}(PK_2, opk', m, \sigma_2) = 1$ and at the same time $\mathsf{TT.Verify}_{\text{static}}(PK_1, opk, opk', \sigma_1) = 1$ this algorithm outputs 1, otherwise 0.

Theorem 5.5 *Let* $\mathsf{TT}_{static} = (\mathsf{TT.KeyGen}_{static}, \mathsf{TT.OTKeyGen}_{static}, \mathsf{TT.Sign}_{static}, \mathsf{TT.Verify}_{static})$ *be a* $(q_{static}, t_{static}, \epsilon_{static})$ *statically secure two-tier signature scheme. Then the above construction yields a* (q, t, ϵ) *adaptively secure two-tier signature scheme provided that* $q = q_{static}$*,* $t = t_{static}$*,* $\epsilon \leq 2\epsilon_{static}$*. Moreover, if* TT_{static} *is strongly secure so is* TT*.*

PROOF. Assume there exists an attacker $\mathcal{A}$ against the adaptively secure two-tier signature scheme. Let $opk_1, \ldots, opk_q$ denote the answers to $\mathcal{A}$'s queries for one-time public keys. Let $m_1, \ldots, m_q$ be the messages queried by $\mathcal{A}$ and $\sigma_1 = (\sigma_{1,1}, \sigma_{2,1}, opk'_1), \ldots, \sigma_q = (\sigma_{1,q}, \sigma_{2,q}, opk'_q)$ the corresponding signatures on the m_i. Suppose $(i^*, m^*, \sigma^* = (\sigma_1^*, \sigma_2^*, opk'^*))$ be the forgery output by $\mathcal{A}$ such that $\mathsf{TT.Verify}(PK, opk_{i^*}, m^*, \sigma^*) = 1$. By assumption, the simulator $\mathcal{B}$ is given the challenge public key PK_a of a statically secure two-tier signature scheme together with q-time access to the oracle $\mathcal{O}_{SK_a,\text{static}}(\cdot)$. Then $\mathcal{B}$ computes some additional key material SK_b, PK_b using $\mathsf{TT.KeyGen}_{\text{static}}(1^\kappa)$. Next $\mathcal{B}$ draws a random coin $b \in \{1, 2\}$ indicating its guess for one of the following two cases. With probability $\geq 1/2$ this guess is correct.

If $b = 1$ (Case 1), $\mathcal{B}$ assumes that $\mathcal{A}$ outputs a forgery $\sigma = (\sigma_1^*, \sigma_2^*, opk'^*)$ that verifies under one-time public key opk_i with $i \in [1; q]$ such that $opk'^* = opk'_i$. In this case, $\mathcal{B}$ sets $PK = (PK_1, PK_2)$ with $PK_1 = PK_b$ and $PK_2 = PK_a$. To simulate the signing oracle, $\mathcal{B}$ uses SK_1, PK_1 to compute fresh one-time keys $(osk_i, opk_i) \leftarrow \mathsf{TT.OTKeyGen}_{\text{static}}(SK_1, PK_1)$ for $i \in [1; q]$. The forger can adaptively query these one-time public keys $opk_1, \ldots, opk_q$. When $\mathcal{A}$ queries message m_i with $i \in [1; q]$, $\mathcal{B}$ sends m_i to the signing oracle $\mathcal{O}_{SK_2,\text{static}}(\cdot)$ to receive a one-time public key opk'_i and a one-time signature $\sigma_{i,2}$ on m_i. Next, $\mathcal{B}$ uses SK_1 and osk_i to compute a one-time signature $\sigma_{i,1} = \mathsf{TT.Sign}_{\text{static}}(SK_1, osk_i, opk'_i)$ on opk'_i. Observe that the values sent in the simulation phase are distributed exactly as in the original attack game. Finally $\mathcal{A}$ outputs a forgery $\sigma = (\sigma_1^*, \sigma_2^*, opk'^*)$ that verifies under one-time public key opk_i with $i \in [1; q]$ such that $opk'^* = opk'_i$. Since by assumption $m^* \neq m_i$ (or $(m^*, \sigma_2^*) \neq (m_i, \sigma_{2,i})$ in the case of strong security), (m^*, σ_2^*) breaks the security of the statically secure two-tier signature scheme.

If $b = 2$ (Case 2), $\mathcal{B}$ assumes that $\mathcal{A}$ outputs a forgery $\sigma = (\sigma_1^*, \sigma_2^*, opk'^*)$ that verifies under one-time public key opk_i with $i \in [1; q]$ such that $opk'^* \neq opk'_i$. In this case, $\mathcal{B}$ sets $PK = (PK_1, PK_2)$ with $PK_1 = PK_a$ and $PK_2 = PK_b$. To simulate the signing oracle, $\mathcal{B}$ uses PK_2 to compute fresh one-time keys $(osk'_i, opk'_i) \leftarrow \mathsf{TT.OTKeyGen}_{\text{static}}(SK_2, PK_2)$ for all $i \in [1; q]$. Next, $\mathcal{B}$ sends $(1, opk'_1), \ldots, (q, opk'_q)$ to $\mathcal{O}_{SK_1,\text{static}}(\cdot)$. As an answer, $\mathcal{B}$ receives q one-time public keys $opk_1, \ldots, opk_q$ and signatures $\sigma_{1,1}, \ldots, \sigma_{q,1}$ on the opk'_i such that $\mathsf{TT.Verify}(PK, opk_i, opk'_i, \sigma_i)$=1. The forger can now adaptively query these one-time public keys $opk_1, \ldots, opk_q$. When $\mathcal{A}$ queries message m_i with $i \in [1; q]$, $\mathcal{B}$ uses SK_2 to compute $\sigma_{i,2} = \mathsf{TT.Sign}_{\text{static}}(SK_2, osk'_i, m_i)$. The final signature is $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, opk'_i)$. Observe that the values sent in the simulation phase are distributed exactly as in the original attack game. Finally $\mathcal{A}$ outputs a forgery $\sigma = (\sigma_1^*, \sigma_2^*, opk'^*)$ that verifies under one-time public key opk_i with $i \in [1; q]$ such that $opk'^* \neq opk'_i$. Since we have that $opk'^* \neq opk'_i$, (opk'^*, σ_1^*) breaks the security of the statically secure two-tier signature scheme. This concludes the proof of Theorem 5.5. □

5.5 Construction of Adaptively Secure Two-Tier Signature Schemes from Chameleon Hash Functions

If we use our statically secure two-tier signature scheme that is based on chameleon hash functions we can further improve the efficiency of the construction. The reason for this is that again opk'_i can be computed from the message m and the signature part σ_2. Therefore opk'_i need not be transmitted to the verifier. Let $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ be a chameleon hash function and $\mathcal{H}$ be a hash function that maps values from the output space of the chameleon hash function to the message space. Then the following construction yields a fully secure two-tier signature scheme $\mathsf{TT} = (\mathsf{TT.KeyGen}, \mathsf{TT.OTKeyGen}, \mathsf{TT.Sign}, \mathsf{TT.Verify})$.

- $\mathsf{TT.KeyGen}(1^\kappa)$: runs $w \leftarrow \mathsf{HF.KeyGen}(1^\kappa)$ and $\mathsf{CH.Gen}(1^\kappa)$ (twice) to obtain two key pairs (SK_1, PK_1) and (SK_2, PK_2). It sets $SK = (SK_2)$ and publishes $PK = (PK_1, PK_2, w)$.

- $\mathsf{TT.OTKeyGen}(SK, PK)$: randomly chooses message $m' \in \mathcal{M}$ and randomness $r', \sigma_1 \in \mathcal{R}$ and computes $t' = \mathsf{CH.Eval}(PK_2, m', r')$ and $opk = \mathsf{CH.Eval}(PK_1, h(t'), \sigma_1)$ with $h(\cdot) := \mathsf{HF.Eval}(w, \cdot)$. The secret one-time key is set to $osk = (m', r', \sigma_1)$. The one-time public key is opk.

- $\mathsf{TT.Sign}(SK, osk, m)$: parses osk as (m', r', σ_1). Then it computes a collision of the chameleon hash function $\sigma_2 = \mathsf{CH.Coll}(SK_2, r', m', m)$ such that $\mathsf{CH.Eval}(PK_2, m, \sigma_2) = \mathsf{CH.Eval}(PK_2, m', r')$. The final signature is $\sigma = (\sigma_1, \sigma_2)$. Observe that σ does not contain $t = \mathsf{CH.Eval}(PK_2, m, \sigma_2)$.

- $\mathsf{TT.Verify}(PK, opk, m, \sigma)$: computes $t = \mathsf{CH.Eval}(PK_2, m, \sigma_2)$ and checks whether it holds that $opk = \mathsf{CH.Eval}(PK_1, h(t), \sigma_1)$. In this case it outputs 1, otherwise 0.

Theorem 5.6 *Let* $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ *be a* $(t', \epsilon_{\mathsf{CH}})$*-collision-resistant chameleon hash function and* h *be a* (t', ϵ_h) *collision-resistant hash function. Then for any polynomial* $q = q(\kappa)$ *the above construction yields a* (q, t, ϵ) *adaptively secure two-tier signature scheme* TT *with* $t \approx t'$ *and* $\epsilon \leq 3/2\epsilon_{\mathsf{CH}} + 3\epsilon_h$*. Moreover if* CH *is strongly secure so is* TT*.*

The proof of Theorem 5.6 is very similar to the proof of Theorem 5.5. The only difference is that we can exploit the special property of our statically secure two-tier signature schemes that is based on chameleon hash functions. First, we only need a single secret key element $SK = SK_2$. Second, each one-time public key can be computed from the message and the corresponding one-time signature. In the above construction of adaptively secure two-tier signatures we exploit this property only once. As a result t (that corresponds to opk' in the previous proof) need not be transferred (as compared to the generic construction of Section 5.4.1). However, when we apply our construction to the $\mathsf{aCMA} \longrightarrow \mathsf{STR\text{-}aCMA}$ transformation we can exploit this property for a second time. Namely, all one-time public keys of our adaptively secure two-tier signature scheme can be computed and need not be transferred to the verifier.

Proof. Let $opk_1, \ldots, opk_q$ denote the answers to $\mathcal{A}$'s queries for one-time public keys. Let $m_1, \ldots, m_q$ be the messages queried by $\mathcal{A}$ and $\sigma_1 = (\sigma_{1,1}, \sigma_{2,1}), \ldots, \sigma_q = (\sigma_{1,q}, \sigma_{2,q})$ the corresponding signatures on the m_i. Let $i^*, m^*, \sigma^* = (\sigma_1^*, \sigma_2^*)$ be the forgery output by $\mathcal{A}$ such that $\mathsf{TT.Verify}(PK, opk_{i^*}, m^*, \sigma^*) = 1$. In the proof we will distinguish three different cases.

- For Type 1 forgeries we have $t^* = \mathsf{CH.Eval}(PK_2, m^*, \sigma_2^*) = t'_{i^*} = \mathsf{CH.Eval}(PK_2, m_{i^*}, \sigma_{2,i^*})$. Any forger that outputs Type 1 forgeries can be used to break the security of the chameleon hash function with the keys (SK_2, PK_2). This is because since $t^* = t_i$, (m^*, σ_2^*) and $(m_i, \sigma_{2,i})$ constitute a collision of the chameleon hash function.
- We also consider Type 2 forgeries where we have that $t^* = \mathsf{CH.Eval}(PK_2, m^*, \sigma_2^*) \neq t'_{i^*} = \mathsf{CH.Eval}(PK_2, m_{i^*}, \sigma_{2,i^*})$ and $h(t'_{i^*}) \neq h(t^*)$. Since we must have the equality $\mathsf{CH.Eval}(PK_1, h(t'_{i^*}), \sigma_{1,i^*}) = opk_{i^*} = \mathsf{CH.Eval}(PK_1, h(t^*), \sigma_1^*)$, $(h(t'_{i^*}), \sigma_{1,i^*})$ and $(h(t^*), \sigma_1^*)$ constitute a collision of the chameleon hash function (SK_1, PK_1).
- In the remaining case, (Type 3 forgeries) it holds that $t^* = \mathsf{CH.Eval}(PK_2, m^*, \sigma_2^*) \neq t'_{i^*} = \mathsf{CH.Eval}(PK_2, m_{i^*}, \sigma_{2,i^*})$ but $h(t'_{i^*}) = h(t^*)$. A Type 3 forger can easily be used to break the security of the underlying hash function when given a challenge key w.

At the beginning, the simulator tosses a random coin $b \in \{1, 2, 3\}$ indicating its guess for Type 1, 2, or 3 forgeries. With probability $\geq 1/3$ this guess is correct. According to its guess the simulator proceeds differently.

If $b = 1$ (Type 1 Forgeries) assume that PK_{CH} is the challenge of the chameleon hash function. The simulator $\mathcal{B}$ at first sets $PK_2 = PK_{\mathsf{CH}}$. Next, it generates $(SK_1, PK_1) \leftarrow \mathsf{CH.Gen}(1^\kappa)$. When $\mathcal{B}$ is asked to output a new one-time key opk_i, $\mathcal{B}$ at first randomly generates message m'_i and randomness r_i and r'_i. Then it computes $t'_i = \mathsf{CH.Eval}(PK_2, m'_i, r'_i)$ and outputs $opk_i = \mathsf{CH.Eval}(PK_1, h(t'_i), r_i)$. When $\mathcal{A}$ queries message m_i, $\mathcal{B}$ generates a random $\sigma_{2,i}$ and computes $t_i = \mathsf{CH.Eval}(PK_2, m_i, \sigma_{2,i})$ and $\sigma_{1,i} = \mathsf{CH.Coll}(SK_1, r_i, h(t'_i), h(t_i))$ using SK_1. Then it outputs $\sigma_i = (\sigma_{1,i}, \sigma_{2,i})$. Eventually $\mathcal{A}$ outputs a forgery $(i^*, m^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$. By assumption we have that $t^* = t_{i^*}$ and $m^* \neq m_{i^*}$. Therefore (m^*, σ_2^*) and $(m_{i^*}, \sigma_{2,i^*})$ constitute a collision of the chameleon hash function. Observe that all values are distributed exactly like in the original attack game.

If $b = 2$ (Type 2 Forgeries) the simulator $\mathcal{B}$ sets $PK_1 = PK_{\mathsf{CH}}$. Next, it generates $(SK_2, PK_2) \leftarrow \mathsf{CH.Gen}(1^\kappa)$. When $\mathcal{B}$ is queried to output a new one-time key opk_i, $\mathcal{B}$ generates random m'_i, r_i, and r'_i. Then it computes $t'_i = \mathsf{CH.Eval}(PK_2, m'_i, r'_i)$ and outputs $opk_i = \mathsf{CH.Eval}(PK_1, h(t'_i), r_i)$. When $\mathcal{A}$ queries message m_i, $\mathcal{B}$ sets $\sigma_{1,i} = r_1$ and computes $\sigma_{2,i} = \mathsf{CH.Coll}(SK_2, r'_i, m'_i, m_i)$ using SK_2. Then it outputs $\sigma_i = (\sigma_{1,i}, \sigma_{2,i})$. Eventually $\mathcal{A}$ outputs a forgery $(i^*, m^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$. By assumption we have that $t^* = \mathsf{CH.Eval}(PK_2, m_i^*, \sigma_2^*) \neq t_{i^*}$ and $h(t^*) \neq h(t_{i^*})$. Therefore $(h(t'_{i^*}), \sigma_{1,i^*})$ and $(h(t^*), \sigma_1^*)$ indeed constitute a collision of the chameleon hash function. Observe that all values are distributed exactly like in the original attack game.

Finally, if $b = 3$ the simulator $\mathcal{B}$ computes $(SK_1, PK_1) \leftarrow \mathsf{CH.Gen}(1^\kappa)$ *and* $(SK_2, PK_2) \leftarrow \mathsf{CH.Gen}(1^\kappa)$. The simulator proceeds exactly as in the description of the scheme except that it embeds the challenge key w' for the hash function in the public key $w := w'$. Eventually $\mathcal{A}$ outputs a forgery with $t^* \neq t_{i^*}$ but $h(t^*) = h(t_{i^*})$. Therefore t^* and t_{i^*} constitute a collision of the hash function. Observe that all values are distributed exactly like in the original attack game.

This concludes the proof of Theorem 5.6.

□

5.6 New Transformation to Strongly Secure Signature Schemes

We can now apply our new construction of two-tier signature schemes to improve the Bellare-Shoup transformation that uses fully secure signature schemes and strongly, adaptively secure two-tier signature schemes to construct strongly secure signature schemes. We exploit the special property of our construction that one-time public keys can be computed given one-time signatures and messages.

5.6.1 Efficient Transformation to Strongly Secure Signature Schemes

Assume that $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ is a strongly collision-resistant chameleon hash function and $\mathcal{H} = (\mathsf{HF.KeyGen}(1^\kappa), \mathsf{HF.Eval}(w, m))$ is a collision resistant hash function. Additionally, assume that $\mathsf{SIG}_{\text{full}} = (\mathsf{SIG.KeyGen}_{\text{full}}, \mathsf{SIG.Sign}_{\text{full}}, \mathsf{SIG.Verify}_{\text{full}})$ is a fully secure signature scheme. Then the following construction yields a strongly secure signature scheme $\mathsf{SIG} = (\mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ that is secure against existential forgeries under adaptive chosen message attacks.

- $\mathsf{SIG.KeyGen}(1^\kappa)$: runs $w \leftarrow \mathsf{HF.KeyGen}(1^\kappa)$ and $\mathsf{CH.Gen}(1^\kappa)$ twice to obtain (SK_1, PK_1) and (SK_2, PK_2). Then it runs $\mathsf{SIG.KeyGen}_{\text{full}}(1^\kappa)$ to obtain $(SK_{\text{full}}, PK_{\text{full}})$. It sets $SK = (SK_{\text{full}}, SK_2)$ and publishes $PK = (PK_{\text{full}}, PK_1, PK_2, w)$.

- $\mathsf{SIG.Sign}(SK, m)$: randomly chooses randomness $m' \in \mathcal{M}$, $r', \sigma_3 \in \mathcal{R}$ and computes $t = \mathsf{CH.Eval}(PK_2, m', r')$, $opk = \mathsf{CH.Eval}(PK_1, h(t), \sigma_3)$, and $\sigma_1 = \mathsf{SIG.Sign}(SK_{\text{full}}, m||opk)$ with $h(\cdot) := \mathsf{HF.Eval}(w, \cdot)$. Then it computes the collision $\sigma_2 = \mathsf{CH.Coll}(SK_2, r', m', \sigma_1)$ $\mathsf{CH.Eval}(PK_2, \sigma_1, \sigma_2) = t$. The final signature is $\sigma = (\sigma_1, (\sigma_2, \sigma_3))$. Observe that σ does not contain neither t nor opk.

- $\mathsf{SIG.Verify}(PK, m, \sigma)$: computes $t = \mathsf{CH.Eval}(PK_2, \sigma_1, \sigma_2)$, $opk = \mathsf{CH.Eval}(PK_1, h(t), \sigma_3)$ and checks whether $\mathsf{SIG.Verify}(PK, m||opk, \sigma_1) = 1$. In this case it outputs 1, otherwise 0.

Theorem 5.7 *Let $\mathsf{CH} = (\mathsf{CH.Gen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ be a $(t', \epsilon_{\mathsf{CH}})$-collision-resistant chameleon hash function, SIG_{full} be a $(q_{full}, t', \epsilon_{full})$ fully secure signature scheme and $\mathcal{H}$ be a (t', ϵ_h) collision resistant hash function. Then the above construction yields a (q, t, ϵ) strongly secure signature scheme SIG with $q = q_{full}$, $t \approx t'$, and $\epsilon \leq 2\epsilon_{\mathsf{CH}} + 4\epsilon_h + 4\epsilon_{full}$.*

PROOF.

Assume there exists an attacker $\mathcal{A}$ that breaks the security of the fully secure signature scheme. Let $(\sigma_{1,1}, (\sigma_{2,1}, \sigma_{3,1})), \ldots, (\sigma_{1,q}, (\sigma_{2,q}, \sigma_{3,q}))$ be the signatures returned as responses to $\mathcal{A}$'s signature queries $m_1, \ldots, m_q$ and let $(\sigma_1^*, (\sigma_2^*, \sigma_3^*))$ be $\mathcal{A}$'s forgery.

In the proof we consider five cases.

- In the first case (Case 1) we have that $m^*||opk^*$ has not been queried before: $m^*||opk^* \notin \{m_1||opk_1, \ldots, m_q||opk_q\}$. Any attacker that outputs such types of forgery can be used to break the security of the underlying fully secure signature scheme.

- In the second case (Case 2) there exists an $i \in [1; q]$ with $m^*||opk^* = m_i||opk_i$ implying $opk^* = opk_i$ and $m^* = m_i$. It holds that $h(t^*) = h(t_i)$, $t^* = \mathsf{CH.Eval}(PK_2, \sigma_1^*, \sigma_2^*) = t_i = \mathsf{CH.Eval}(PK_2, \sigma_{1,i}, \sigma_{2,i})$, and $(\sigma_1^*, \sigma_2^*) \neq (\sigma_{1,i}, \sigma_{2,i})$. Thus (σ_1^*, σ_2^*) and $(\sigma_{1,i}, \sigma_{2,i})$ constitute a collision in the chameleon hash function with the key material (SK_2, PK_2).

- In the third case (Case 3) there exists an $i \in [1; q]$ with $m^*||opk^* = m_i||opk_i$ implying $opk^* = opk_i$ and $m^* = m_i$. It holds that $h(t^*) = h(t_i)$, $t^* = \mathsf{CH.Eval}(PK_2, \sigma_1^*, \sigma_2^*) = t_i = \mathsf{CH.Eval}(PK_2, \sigma_{1,i}, \sigma_{2,i})$, and $(\sigma_1^*, \sigma_2^*) = (\sigma_{1,i}, \sigma_{2,i})$. We have that $\sigma_3^* \neq \sigma_{3,i}$ since otherwise σ^* would not be a valid forgery. Therefore σ_3^* is a new one-time signature on message m_i under one-time public key opk_i.Thus $(h(t_i), \sigma_3^*)$ and $(h(t_i), \sigma_{3,i})$ break the strong unforgeability of the chameleon hash function with the keys (SK_1, PK_1).

- In the fourth case (Case 4) there exists an $i \in [1; q]$ with $m^*||opk^* = m_i||opk_i$ implying $opk^* = opk_i$ and $m^* = m_i$. It holds that $t^* = \mathsf{CH.Eval}(PK_2, \sigma_1^*, \sigma_2^*) \neq t_i = \mathsf{CH.Eval}(PK_2, \sigma_{1,i}, \sigma_{2,i})$ but $h(t^*) = h(t_i)$. This breaks the security of the hash function.

- In the last case (Case 5) there exists an $i \in [1;q]$ with $m^*||opk^* = m_i||opk_i$ implying $opk^* = opk_i$ and $m^* = m_i$. It holds that $t^* = \mathsf{CH.Eval}(PK_2, \sigma_1^*, \sigma_2^*) = t_i = \mathsf{CH.Eval}(PK_2, \sigma_{1,i}, \sigma_{2,i})$ and $h(t^*) \neq h(t_i)$. Since $opk^* = opk_i$ we have that $opk^* = \mathsf{CH.Eval}(PK_1, h(t^*), \sigma_3^*) = opk_i = \mathsf{CH.Eval}(PK_1, h(t_i), \sigma_{3,i})$. This breaks the security of the chameleon hash function with the keys (SK_1, PK_1).

Observe that the above cases cover all possible forgeries $\mathcal{A}$ might output. Let us now show how the simulator $\mathcal{B}$ can set up the input parameters and answer the signature queries in each of the cases. At the beginning of the simulation, $\mathcal{B}$ tosses a random coin $b \in \{1, 2, 3, 4\}$ indicating its guess for Case 1 ($b = 1$), Case 2 ($b = 2$), Case 3 or 5 ($b = 3$), or Case 4 ($b = 4$). With probability $\geq 1/4$ this guess is correct. According to its guess the simulator proceeds differently.

- In Case 1 the simulator sets $PK_{\text{full}} = PK'_{\text{full}}$, where PK'_{full} is the challenge public key received from the full security attack game. All other keys are generated exactly as in the description of the construction. Using oracle $O(SK'_{\text{full}}, \cdot)$ and SK_2, $\mathcal{B}$ can easily answer all signature queries. Observe that all transmitted values are exactly distributed as in the original attack game.

- In Case 2 the simulator sets $PK_2 = PK'_{\mathsf{CH}}$, where PK'_{CH} is the challenge public key received from the chameleon hash security game. All other keys are generated exactly as in the description of the construction. Since SK_2 is unknown, $\mathcal{B}$ cannot compute collisions of the chameleon hash function PK_2 as described in the construction. However, $\mathcal{B}$ can now compute collisions for PK_1. To compute a signature on query m_i, $\mathcal{B}$ randomly chooses message m', and randomness σ_2 and r' and computes $t = \mathsf{CH.Eval}(PK_2, m', \sigma_2)$, $opk = \mathsf{CH.Eval}(PK_1, h(t), r')$, and $\sigma_1 = \mathsf{SIG.Sign}(SK_{\text{full}}, m||opk)$ with $h(\cdot) := \mathsf{HF.Eval}(w, \cdot)$. Then it computes $t' = \mathsf{CH.Eval}(PK_2, \sigma_1, \sigma_2)$ and the collision $\sigma_3 = \mathsf{CH.Coll}(SK_1, r', h(t), h(t'))$ such that $\mathsf{CH.Eval}(PK_3, h(t'), \sigma_3) = opk$. The final signature is $\sigma = (\sigma_1, (\sigma_2, \sigma_3))$. Observe that all transmitted values are exactly distributed as in the original attack game.

- In Case 3 and Case 5 the simulator sets $PK_1 = PK'_{\mathsf{CH}}$, where PK'_{CH} is the challenge public key received from the chameleon hash security game. All other keys are generated exactly as in the description of the construction. Using SK_{full} and SK_2, $\mathcal{B}$ can answer all signature queries as described in the construction. Observe that all transmitted values are exactly distributed as in the original attack game.

- In Case 4 the simulator sets $w = w'$ where w' is the challenge key received from the attack game of the cryptographic hash function. All other keys are generated exactly as in the description of the construction. Observe that all transmitted values are exactly distributed as in the original attack game.

□

In contrast to the Huang *et al.* transformation, our final construction tightly reduces to the signature scheme *and* the chameleon hash function. Let us now compare our construction with the one presented in [13].

5.6.2 Comparison with the Bellare-Shoup Construction

The Bellare-Shoup construction is derived from three move identification protocols via the Fiat-Shamir heuristic except that the hash function is not modeled as a random oracle but solely as a collision-resistant hash function. Let C_M be the first, C_H the second, and R the third message sent in such an identification protocol. The one-time signature only consists of R – a single 160 bit exponent. However, in cascaded constructions the one-time public key C_M must also be transferred and in identification protocols C_M is a group element. For comparison, recall the well-known zero-knowledge protocol for proving knowledge of a discrete logarithm [94]. Here, the Bellare-Shoup construction requires to transmit one exponent and a group element C_M where C_M is approximately 1024 bits. Our construction, when using a discrete logarithm based chameleon hash function, requires only two exponents to be transferred to the verifier amounting to only 320 bits! We stress that all constructions presented in [13] require at least a single group element C_M and some additional values. The most efficient construction is based on the Schnorr protocol [94] and consists of a group element and a single exponent. However, the security of the resulting two-tier signature is based on the 'one-more' discrete logarithm assumption which is weaker than the mere discrete logarithm assumption. Therefore, when used to construct strongly secure signature schemes our instantiation is not only more efficient than the most efficient instantiation of [13] but also based on weaker security assumptions.

Chapter 6

Ring Signature Schemes

The CDH assumption became practical for standard model signature schemes with the introduction of bilinear pairings into cryptography. In 2005, Waters showed the existence of a hash-and-sign signature scheme that is secure under the CDH assumption in the standard model [102]. Since then several signature schemes, including ring signature schemes [95], sequential aggregate signature schemes, multisignature schemes, and verifiably encrypted signature schemes [74] have been proposed that are secure in the standard model. In this chapter we develop a new and efficient ring signature schemes without random oracles that is solely based on the CDH assumption in symmetric bilinear groups.

6.0.3 Related Work on Ring Signature Schemes

The first (explicit) ring signature scheme by Rivest, Shamir and Tauman [91] was proven secure in the random oracle/ideal cipher model [7, 41]. Since then, several ring signature schemes have been proposed in the random oracle model. However, today only a handful of ring signature schemes proven secure without random oracles exist.

While the scheme of Chow *et al.* [40] published in 2006 provides unconditional anonymity, unforgeability is based on a new strong assumption that is given without any evidence for its validity. In the same year, Bender, Katz and Morselli proposed a ring signature scheme based on trapdoor permutations, but since it uses generic ZAPs for NP it is impractical for real world applications [14]. In the same work the authors also presented two 2-user ring signature schemes without random oracles that are secure under the CDH and the LRSW assumption. Disadvantageously, these schemes only allow to issue signatures on rings of maximal size 2. This is security critical since in a ring signature scheme the provided level of signer anonymity is primarily determined by the number of ring members. Thus, dependent on the application and his requirements on an appropriate security level *the user* should decide on the size of the ring. In 2007, Shacham and Waters presented a ring signature scheme [95] that is full key-exposure anonymous, a strong security notion stemming from [14], under the Subgroup Decision assumption [20]. Unfortunately, this assumption relies on groups with composite order such that a trusted setup procedure is necessary in the setup phase. Also, the representation of group elements is rather large (about 1024 bits). Unforgeability is based on the CDH assumption and the signature size is $2n+2$ group elements, where n is the size of the ring. In the same year, Boyen presented a new signature scheme with perfect anonymity [24]. Unforgeability of this scheme is based on a new complexity assumption, the Pluri-SDH assumption, while evidence for its usefulness is provided by a security analysis in the generic group model. The signature size consist of n group elements and n integers (of 160 bits) while each public key consists of at least three group elements. Recently, Chandran, Groth and Sahai proposed a new signature scheme with perfect anonymity that is secure under the Subgroup Decision assumption and the Strong Diffie-Hellman assumption [36]. Since the above remarks concerning the trusted setup of [95] also hold here, the authors present two variants of their ring signature scheme. The second variant accounts for maliciously generated common reference strings by heuristically guaranteeing (by using a factorization algorithm) that the composite group order output by the setup algorithm is hard to factor.

Except for the schemes by Chandran *et al.* [36] and Dodis *et al.* [47] (in the random oracle model), all existing ring signature schemes offer signature sizes that are at least linear in the ring size. Both, [36] and [47] provide better (asymptotic) efficiency when several messages are signed using the same ring.

6.0.4 Contribution

In this chapter we present a new ring signature scheme for rings of arbitrary size. Anonymity is perfect, unforgeability solely relies on the CDH assumption in bilinear groups. Security is proven in the fully untrusted common reference string model. The signature size is very small, accounting for only $n + 1$ group elements. Since we use programmable hash functions [64], a drawback of our scheme is that we require relatively large global parameters, consisting of around 160 group elements. However, these parameters can be re-used for all instantiations of the scheme that use the same bilinear group. Advantageously, in our ring signature scheme, each public key consists of a *single* group element such that for large groups (e.g. $>$1000), the public parameters only account for a small portion of the data required for signature generation and verification. Finally we provide a new proof technique for Waters-like signature schemes which is very clean and compact at the same time. The main drawback of our scheme is that it only provides security under chosen subring attacks, where the attacker is not allowed to query secret keys of honest ring members. We stress that our ring signature scheme is much more practical than the CDH-based scheme by Bender, Katz, and Morselli that is also secure under the CDH assumption. First, our scheme can be used to sign messages for rings of arbitrary length, not only for 2-user rings. Second, in our scheme a public key contains only a single group element whereas in the Bender *et al.* scheme a public key consists of a complete group hash function.

6.1 A New CDH-based Ring Signature Scheme

In the following, we briefly recall some of the basic properties of bilinear groups. Definitions 6.1 and 6.3 help to support the intuition behind our security proof. In [23], Boneh, Mironov and Shoup use a similar approach to describe their tree-based signature scheme. However, in contrast to [23], we focus on proving security under the classical CDH assumption, where the challenge and the solution consist of elements from a single group $\mathbb{G}$. We therefore concentrate on symmetric bilinear groups. We stress that, after some minor modifications, we can base our signature schemes on asymmetric bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with an efficient homomorphism $\Phi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. However, security is then based on the co-CDH assumption.

Definition 6.1 (Secure bilinear map) *Let $l = l(\kappa)$ be a polynomial. Let $B = (\mathbb{G}, g, \mathbb{G}_T, p, e)$ be a symmetric bilinear group with $|p|_2 = l$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is (t, ϵ)-secure if for all t-time adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[e(g, g') = e(h, \mathcal{A}(g, g', h)) \mid \ g, g', h \in_R \mathbb{G}, h \neq 1_{\mathbb{G}}\right] \leq \epsilon,$$

where the probability is taken over the random coin tosses of $\mathcal{A}$ and the random choices of g, g', and h.

One can easily see that in symmetric bilinear groups, breaking the security of a bilinear map is equivalent to breaking the CDH assumption.

Lemma 6.2 *Let $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ be a symmetric bilinear group. Then, e is (t, ϵ)-secure if and only if the (t, ϵ)-CDH assumption holds in $\mathbb{G}$.*

Proof. By contradiction. Let $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ be our bilinear group. First, assume attacker $\mathcal{A}$ can break the security of the bilinear map in time t with advantage at least ϵ. Then, algorithm $\mathcal{B}$ can solve the CDH assumption in $\mathbb{G}$ in time t with advantage ϵ by using $\mathcal{A}$ as a black-box. Let $\bar{g}, \bar{g}^a, \bar{g}^b$ be $\mathcal{B}$'s CDH challenge in group $\mathbb{G}$. $\mathcal{B}$ sets $\tilde{g} = \bar{g}^a$, $\tilde{g}' = \bar{g}^b$, and $\tilde{h} = \bar{g}$ and runs attacker $\mathcal{A}$ on $(\tilde{g}, \tilde{g}', \tilde{h})$. As a result, $\mathcal{A}$ outputs $\tilde{h}'$ such that $e(\tilde{g}, \tilde{g}') = e(\tilde{h}, \tilde{h}')$. Since equivalently $e(\bar{g}^a, \bar{g}^b) = e(\bar{g}, \tilde{h}')$, $\tilde{h}'$ is a solution to the CDH problem.
Now, assume adversary $\mathcal{A}$ (t, ϵ)-breaks the CDH assumption in $\mathbb{G}$. Let $\tilde{g}, \tilde{g}', \tilde{h} \in \mathbb{G}$, $\tilde{h} \neq 1_{\mathbb{G}}$ be $\mathcal{B}$'s challenge against the security of the bilinear map. Since $\tilde{h}$ is a generator, there exist $a, b \in \mathbb{Z}_p$ such that $\tilde{h}^a = \tilde{g}$, and $\tilde{h}^b = \tilde{g}'$. $\mathcal{B}$ runs $\mathcal{A}$ on $\tilde{h}, \tilde{g}, \tilde{g}'$. Because $\mathcal{A}$ outputs $\tilde{h}^{ab}$, we have that $e(\tilde{g}, \tilde{g}') = e(\tilde{h}, \tilde{h}^{ab})$, and thus $\mathcal{A}$'s output is a correct solution to $\mathcal{B}$'s challenge. $\square$

Let again $B = (\mathbb{G}, g, \mathbb{G}_T, p, e)$ be a bilinear group with a secure bilinear map e.

Definition 6.3 (Collision generator for bilinear groups) *A collision generator for e is a polynomial time algorithm that on input two elements $g, h \in \mathbb{G}$ outputs a collision $(g', h') \in \mathbb{G}$ such that*

$$e(g, g') = e(h, h').$$

For symmetric pairings there exists an efficient collision generator that can output *all* possible collisions: given g, h randomly choose $r \in \mathbb{Z}_p$ and compute $g' = h^r$ and $h' = g^r$.

6.2 Efficient Ring Signature Scheme RSIG

In this section we present our ring signature scheme RSIG that allows for very short public keys and signatures. In RSIG, the global parameters consist of $l+2$ random elements $h, u_0, u_1, \ldots, u_l \in \mathbb{G}$ that give rise to a group hash function $u(m) = u_0 \prod_{j=1}^{l} u_j^{m_j}$ and a symmetric bilinear group $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ with a secure bilinear map.

RSIG.KeyGen(1^κ)**.** Each user i chooses a random element $x_i \in \mathbb{Z}_p$ as his secret key SK_i. The corresponding public key is $PK_i = g^{x_i}$.

RSIG.Sign$(PK_1, \ldots, PK_n, SK_t, m)$**.** Given a ring of n public keys, the holder of secret key SK_t with $t \in \{1, \ldots, n\}$ can sign a message $m \in \{0,1\}^l$ in the following way: for all $i \in \{1, \ldots, n+1\} \setminus \{t\}$ he chooses $r_i \in_R \mathbb{Z}_p$ and sets

$$s_i = g^{r_i}.$$

Then, he computes

$$s_t = \left(h \cdot \prod_{\substack{i=1 \\ i \neq t}}^{n} PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^{l} u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_t} .$$

The final signature is $\sigma = (s_1, \ldots, s_{n+1})$.

RSIG.Verify$(PK_1, \ldots, PK_n, m, \sigma)$**.** Given a set of n public keys, a message m, and a ring signature $\sigma = (s_1, \ldots, s_{n+1})$, verify the following equation:

$$\prod_{i=1}^{n} e(s_i, PK_i) \cdot e\left(s_{n+1}, u_0 \prod_{j=1}^{l} u_j^{m_j} \right) \stackrel{?}{=} e(g, h) .$$

6.3 Security

In this section, we show that RSIG provides ring unforgeability and perfect ring anonymity according to Definition 2.33 and 2.34 (correctness can easily be verified by inspection).

6.3.1 Ring Unforgeability

Theorem 6.4 *Suppose the $(t_{CDH}, \epsilon_{CDH})$-CDH assumption holds in the group $\mathbb{G}$. Then the ring signature scheme* RSIG *is (t, ϵ, q)-secure against chosen subring attacks provided that*

$$\epsilon \leq \epsilon_{CDH}/P_{q,l}, \quad t \approx t_{CDH}.$$

Proof. By contradiction. Assume there exists an adversary $\mathcal{A}$ that breaks the security of the ring signature scheme in time t with probability ϵ after q signature queries. Then, one can construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a black box to solve the CDH assumption. We assume that $\mathcal{B}$ is given a random challenge for the CDH-problem: $\left(\bar{g}, \bar{g}^a, \bar{g}^b\right) \in \mathbb{G}^3$. The main idea behind our proof is the following. Recall Definition 6.3 and Lemma 6.2. Given two group elements $g, h \in \mathbb{G}$, it is easy to generate all pairs $(g', h') \in \mathbb{G}^2$ such that $e(g, g') = e(h, h')$.

On the other hand, given three group elements g, g', h, the problem of finding a corresponding h' is as hard as solving the CDH problem. Our aim is to transfer this situation to the unforgeability game of our ring signature scheme: the simulator has to choose the input parameters for the attacker such that answering signature queries is as easy as computing collisions and computing a forgery is as hard as breaking the CDH assumption.

Let $\Pr[S_i]$ denote the success probability for an attacker to successfully forge signatures in Game i.

Game$_0$. This is the original attack game. By assumption, attacker $\mathcal{A}$ (t, ϵ, q)-breaks RSIG when interacting with the challenger. We have,

$$\Pr[S_0] = \epsilon \tag{6.1}$$

Game$_1$. This game is like the previous one except that $\mathcal{B}$ constructs the global parameters and the secret and public keys using the algorithms of the programmable hash function and the CDH challenge. First, $\mathcal{B}$ randomly chooses: n elements $x_i \in_R \mathbb{Z}_p$ for $i = 1, \ldots, n$, $l+1$ elements $a'_0, a_1, \ldots, a_l \in_R \{-1, 0, 1\}$, and $l+1$ elements $b_0, b_1, \ldots, b_l \in_R \mathbb{Z}_p$. Let $a_0 = a'_0 - 1$. Then, for all $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, l\}$ $\mathcal{B}$ computes

$$g := \bar{g}^a,\ h := \bar{g}^b,\ PK_i := \bar{g}^{x_i},\ u_j := h^{a_j}\bar{g}^{b_j}.$$

using the CDH challenge. Due to the properties of the multi-generator programmable hash function the distribution of the so computed values is equal to the distribution in the previous game. Thus,

$$\Pr[S_1] = \Pr[S_0]\ . \tag{6.2}$$

Game$_2$. Now, $\mathcal{B}$ simulates the challenger in the attack game by answering $\mathcal{A}$'s signature queries (m_j, R_j, e_j). For convenience, let $a(m) = a_0 + \sum_{i=1}^{l} a_i m_i$ and $b(m) = b_0 + \sum_{i=1}^{l} b_i m_i$. Let $I[j] \subset \{1, \ldots, n\}$ be the set of all indices $i \in \{1, \ldots, n\}$ such that PK_i is a component of R_j. When receiving a signature query, $\mathcal{B}$ at first tests whether $a(m_j) = 0$. In this case, $\mathcal{B}$ outputs the failure signal $\mathtt{F}_1$ and aborts. Otherwise $\mathcal{B}$ chooses $r \in_R \mathbb{Z}_p$ and computes a collision $(d_{\bar{g}}, d_h)$ as $d_{\bar{g}} = h^r$ and $d_h = \bar{g}^r$. Note that by construction $e(d_{\bar{g}}, \bar{g}) = e(d_h, h)$.
The aim of $\mathcal{B}$ is to compute $s_{n+1} \in \mathbb{G}$ and $|I[j]|$ values $s_i \in \mathbb{G}$ (for all $i \in I[j]$) such that

$$\prod_{i \in I[j]} e(s_i, PK_i) \cdot e(s_{n+1}, u(m_j)) = e(g, h)$$

or equivalently

$$e\left(s_{n+1}^{b(m_j)} \cdot \prod_{i \in I[j]} s_i^{x_i}, \bar{g}\right) = e\left(g s_{n+1}^{-a(m_j)}, h\right).$$

In the next step, $\mathcal{B}$ chooses $y \in_R I[j]$ and for all $i \in I[j] \setminus \{y\}$ $s_i \in_R \mathbb{G}$. The values s_y and s_{n+1} are computed in the following way:

$$s_{n+1} = \left(g d_h^{-1}\right)^{1/a(m_j)}, \; s_y = \left(d_{\bar{g}} \cdot s_{n+1}^{-b(m_j)} \cdot \prod_{i \in I[j]\setminus\{y\}} s_i^{-x_i} \right)^{1/x_y} .$$

$\mathcal{B}$ outputs the ring signature $\sigma = (s_1, s_2, \ldots, s_n, s_{n+1})$. The probability for $\mathcal{B}$ to win this game is

$$\Pr[S_2] = \Pr[S_1 \wedge \bar{\mathtt{F}_1}] \; . \tag{6.3}$$

Game₃. In this game $\mathcal{B}$ uses $\mathcal{A}$'s forgery $(m^*, R^*, \sigma^* = (s_1^*, s_2^*, \ldots, s_{n+1}^*))$ to break the CDH assumption. Adversary $\mathcal{B}$ at first checks whether $a(m^*) = 0$. If not, $\mathcal{B}$ outputs the failure signal $\mathtt{F}_2$ and aborts. We get that

$$\Pr[S_3] = \Pr[S_2 \wedge \bar{\mathtt{F}_2}] \; . \tag{6.4}$$

Otherwise, $\mathcal{B}$ computes the solution to the CDH problem as follows. Since $a(m^*) = 0$, we get that

$$e\left((s_{n+1}^*)^{b(m^*)} \cdot \prod_{i \in I[*]} (s_i^*)^{x_i}, \bar{g} \right) = e(g, h) \;\Leftrightarrow\; \bar{g}^{ab} = (s_{n+1}^*)^{b(m^*)} \cdot \prod_{i \in I[*]} (s_i^*)^{x_i}$$

what constitutes a solution to the CDH challenge.
We finally have

$$\Pr[S_3] = \epsilon_{\text{CDH}} \; . \tag{6.5}$$

Now, let us analyze the probabilities for an abort, i.e. for one of the events $\mathtt{F}_1$ or $\mathtt{F}_2$ to occur. Surely, the probability that both failure events do not occur depends on the number of signature queries q and the bit size l of the messages. Since, $u(m)$ is generated by the multi-generator programmable hash function as defined in Sect. 2.9.1, we can directly apply the results from [64] to show that

$$\Pr[\bar{\mathtt{F}_1} \wedge \bar{\mathtt{F}_2}] \geq P_{q,l} \; .$$

Putting (6.1-6.5) together, we get that

$$\epsilon_{\text{CDH}} = \Pr[S_0 \wedge \bar{\mathtt{F}_1} \wedge \bar{\mathtt{F}_2}] = \Pr[S_0 | \bar{\mathtt{F}_1} \wedge \bar{\mathtt{F}_2}] \cdot \Pr[\bar{\mathtt{F}_1} \wedge \bar{\mathtt{F}_2}] \geq \epsilon \cdot P_{q,l}$$

which proves Theorem 6.4.

□

6.3.2 Ring Anonymity

Theorem 6.5 *The ring signature scheme* RSIG *is perfectly secure.*

We give an information theoretic argument. Given a ring signature, we have to show that each ring member could possibly have created it. Consider a ring signature on message m, that has been created using SK_z. We show that with the same probability it could have been created using SK_y with $y \neq z$. The proof is straight-forward.

PROOF. Fix an arbitrary ring R of n public keys and choose two indices $y, z \in_R \{1, \ldots, n\}$. Next, fix a random $m \in \{0,1\}^l$ and $n-1$ random values r_i with $i \in \{1, \ldots, n+1\} \setminus \{y, z\}$. We show that for any r_y there exists an r_z such that the final signatures generated by RSIG.Sign with either (r_y, SK_z) or (r_z, SK_y) are equal. Since $\mathbb{G}$ is a cyclic group with prime order p, there exists $t \in \mathbb{Z}_p$ and $b(M) = b_0 + \sum_{i=1}^{l} M_i b_i$ with $b_i \in \mathbb{Z}_p$ such that $h = g^t$ and $u(M) = g^{b(M)}$ for all $M \in \{1, \ldots, n\}$.
Let the ring signature consist of all $s_i = g^{r_i}$ with $i \in \{1, \ldots, n\} \setminus \{y, z\}$. Then, the remaining s_y, s_z are computed using SK_z and the RSIG.Sign algorithm as

$$s_y = g^{r_y},\ s_z = \left(h \cdot \prod_{\substack{i=1\\ i\neq z}}^{n} PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^{l} u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_z} .$$

Now, let $r_z = \frac{t - \sum_{i=1, i\neq z}^{n} r_i x_i - r_{n+1} b(m)}{x_z}$. Using SK_y we get $s_z = g^{r_z}$ and

$$s_y = \left(h \cdot \prod_{\substack{i=1\\ i\neq y}}^{n} PK_i^{-r_i} \cdot \left(u_0 \prod_{j=1}^{l} u_j^{m_j} \right)^{-r_{n+1}} \right)^{1/x_y} = g^{r_y}$$

with $r_y = \frac{t - \sum_{i=1, i\neq y}^{n} r_i x_i - r_{n+1} b(m)}{x_y}$ what concludes the proof of Theorem 6.5. □

Table 6.1: Comparison of the Waters signature scheme and $\mathcal{S}$ and $\mathcal{S}_0$. Unless not stated otherwise, all values are elements of $\mathbb{G}$. We set $u(m) = u_0 \prod_{i=1}^{l} u_i^{m_i}$ and $x(m) = x_0 + \sum_{i=1}^{l} x_i m_i$.

	Waters [102]	$\mathcal{S}$	$\mathcal{S}_0$
publ. params.	$g_0, h, u_0, \ldots, u_l$	$g, h, u_0, \ldots, u_l$	g_0, g, h
SK	h^x	$x \in \mathbb{Z}_p$	$x_0, \ldots, x_l \in \mathbb{Z}_p$
PK	$g = g_0^x$	$g_0 = g^x$	$u_0 = g^{x_0}, \ldots, u_l = g^{x_l}$
s_1	$h^x \cdot (u(m))^r$	$(h \cdot (u(m))^r)^{\frac{1}{x}}$	g^{-r}
s_2	g_0^{-r}	g^{-r}	$(hg_0^r)^{\frac{1}{x(m)}}$
verification	$e(s_1, g_0) \cdot e(s_2, u(m)) \overset{?}{=} e(g, h)$		

6.3.3 Digital Signature Schemes

Our new proof technique can also be applied to other CDH based signature schemes. For example, we can surprisingly easy obtain as a special case ($n = 1$) of our ring signature scheme a variant $\mathcal{S}$ of the Waters signature scheme that has distinct setup and sign algorithms but the same verification equation. We briefly compare it with the original scheme by Waters in Table 6.1. For completeness, we also describe a third variant $\mathcal{S}_0$ where the group hash function constitutes the public key of the user. Both schemes can easily be proven secure under the standard notion of security for digital signatures by Goldwasser, Micali and Rivest [62] by adapting the proof of Theorem 6.4.

Chapter 7

Conclusion

The previous chapters presented four results:

1. a new SDH-based signature scheme together with an improvement of the SRSA-based signature scheme by Naccache, Pointcheval, and Stern;
2. the first tight security proofs for two large classes of signature schemes;
3. new security notions, transformations, and efficient constructions of two-tier signature schemes from chameleon hash functions;
4. a new and very efficient ring signature scheme which is solely secure under the CDH assumption in bilinear groups.

Despite encompassing several rather theoretical analyses, all these results aim at a very practical goal – increasing the efficiency of state-of-the art signature schemes while relying on weak security assumptions. Future research can continue this avenue. Concrete open (and arguably promising) research problems are:

It is worthwhile to find further results in the spirit of 2). In particular, it is interesting to find tight security proofs for the CDH-based signature scheme by Waters [102] and the very efficient SRSA-based and SDH-based signature schemes by Hofheinz and Kiltz [64]. Complementary or alternatively, it may be interesting to devise a framework that allows to show whether tight security proofs can exist at all for a given signature scheme. With respect to 4), it seems interesting to analyze how one can extend the proposed ring signature scheme to guarantee security under stronger unforgeability notions where the attacker can also corrupt secret keys. A straight-forward approach is to utilize carefully designed and efficient non-interactive zero-knowledge proofs. The main challenge is to have these proofs rely on very weak security assumptions, preferably the CDH assumptions or weaker, such that the resulting scheme still only relies on the CDH assumption. From a practical point of view it is interesting to find new and promising applications of two-tier signature schemes. Two-tier signature schemes are particularly useful in scenarios where a party repeatedly issues one-time signatures for freshly generated one-time keys.

Besides the above research directions, there are of course several important long-standing open problems in the realm of digital signature schemes. Probably the most interesting (and challenging) is to design an efficient hash-and-sign signature scheme which is solely secure under the factoring or the DL assumption.

Bibliography

[1] Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 36–53, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.

[2] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.

[3] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT*, pages 480–494, 1997.

[4] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT*, pages 236–250, 1998.

[5] Mihir Bellare and Silvio Micali. How to sign given any trapdoor permutation. *J. ACM*, 39(1):214–233, 1992.

[6] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.

[7] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[8] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[9] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996.

[10] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 1997.

[11] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay [100], pages 409–426.

[12] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.

[13] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Okamoto and Wang [86], pages 201–216.

[14] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Halevi and Rabin [63], pages 60–79.

[15] Daniel J. Bernstein. Proving tight security for Rabin-Williams signatures. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2008.

[16] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Cachin and Camenisch [27], pages 56–73.

[17] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[18] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Franklin [55], pages 41–55.

[19] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[20] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[21] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer, Berlin, Germany.

[22] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[23] Dan Boneh, Ilya Mironov, and Victor Shoup. A secure signature scheme from bilinear maps. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 98–110. Springer, 2003.

[24] Xavier Boyen. Mesh signatures. In Naor [81], pages 210–227.

[25] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086, February 2010. `http://eprint.iacr.org/`, version from 13th February 2010.

[26] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 132–145. ACM, 2004.

[27] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[28] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *demix* anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 21–30. ACM, 2002.

[29] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In Naor [81], pages 246–263.

[30] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[31] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Franklin [55], pages 56–72.

[32] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[33] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.

[34] Dario Catalano and Rosario Gennaro. Cramer-damgård signatures revisited: Efficient flat-tree signatures based on factoring. *Theor. Comput. Sci.*, 370(1-3):186–200, 2007.

[35] Dario Catalano, Mario Di Raimondo, Dario Fiore, and Rosario Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2008.

[36] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434. Springer, 2007.

[37] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[38] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[39] Benoît Chevallier-Mames and Marc Joye. A practical and tightly secure signature scheme without hash function. In Masayuki Abe, editor, *CT-RSA*, volume 4377 of *Lecture Notes in Computer Science*, pages 339–356. Springer, 2007.

[40] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS*, pages 297–302. ACM, 2006.

[41] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In Wagner [101], pages 1–20.

[42] Ronald Cramer and Ivan Damgård. Escure signature schemes based on interactive protocols. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 297–310, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Berlin, Germany.

[43] Ronald Cramer and Ivan Damgård. New generation of secure and practical RSA-based signatures. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 173–185, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Berlin, Germany.

[44] Ronald Cramer, Ivan Damgård, and Torben P. Pedersen. Efficient and provable security amplifications. In T. Mark A. Lomas, editor, *Security Protocols Workshop*, volume 1189 of *Lecture Notes in Computer Science*, pages 101–109. Springer, 1996.

[45] Ronald Cramer and Victor Shoup. Signature schemes based on the Strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[46] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[47] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Cachin and Camenisch [27], pages 609–626.

[48] Cynthia Dwork and Moni Naor. An efficient existentially unforgeable signature scheme and its applications. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 234–246, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Berlin, Germany.

[49] ECRYPT2 NoE. ECRYPT2 Yearly Report on Algorithms and Keysizes (2009-2010). Revision 1.0. Technical report, March 2010.

[50] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1985. Springer, Berlin, Germany.

[51] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *J. Cryptology*, 9(1):35–67, 1996.

[52] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.

[53] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Odlyzko [83], pages 186–194.

[54] Marc Fischlin. The Cramer-Shoup Strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2003.

[55] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.

[56] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT*, pages 123–139, 1999.

[57] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[58] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007.

[59] Oded Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In Odlyzko [83], pages 104–110.

[60] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–. IEEE Computer Society, 2003.

[61] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem (abstract) (impromptu talk). In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, page 467, Santa Barbara, CA, USA, August 19–23, 1985. Springer, Berlin, Germany.

[62] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[63] Shai Halevi and Tal Rabin, editors. *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*. Springer, 2006.

[64] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In Wagner [101], pages 21–38.

[65] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.

[66] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 654–670. Springer, 2009.

[67] Qiong Huang, Duncan S. Wong, Jin Li, and Yiming Zhao. Generic transformation from weakly to strongly unforgeable signatures. *J. Comput. Sci. Technol.*, 23(2):240–252, 2008.

[68] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 2009.

[69] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.

[70] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*. The Internet Society, 2000.

[71] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International Computer Science Laboratory, October 1979.

[72] Gregor Leander and Andy Rupp. On the equivalence of rsa and factoring regarding generic ring algorithms. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 241–251. Springer, 2006.

[73] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005.

[74] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Vaudenay [100], pages 465–485.

[75] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999.

[76] Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

[77] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[78] Ilya Mironov. Collision-resistant no more: Hash-and-sign paradigm revisited. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2006.

[79] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In Dongho Won, editor, *ICISC*, volume 2015 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2000.

[80] David Naccache, David Pointcheval, and Jacques Stern. Twin signatures: an alternative to the hash-and-sign paradigm. In *ACM Conference on Computer and Communications Security*, pages 20–27, 2001.

[81] Moni Naor, editor. *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques,*

Barcelona, Spain, May 20-24, 2007, Proceedings, volume 4515 of *Lecture Notes in Computer Science*. Springer, 2007.

[82] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

[83] Andrew M. Odlyzko, editor. *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*. Springer, 1987.

[84] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Germany.

[85] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Halevi and Rabin [63], pages 80–99.

[86] Tatsuaki Okamoto and Xiaoyun Wang, editors. *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*. Springer, 2007.

[87] Adrian Perrig. The BiBa one-time signature and broadcast authentication protocol. In *ACM Conference on Computer and Communications Security*, pages 28–37, 2001.

[88] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.

[89] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 144–153. Springer, 2002.

[90] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[91] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[92] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.

[93] Barkley Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):211–232, 1941.

[94] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[95] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Okamoto and Wang [86], pages 166–180.

[96] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2001.

[97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Germany.

[98] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, manuscript, nov. 30, 2004. revised version from jan. 18, 2006., 2004.

[99] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, pages 334–345, 1998.

[100] Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[101] David Wagner, editor. *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*. Springer, 2008.

[102] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[103] Huafei Zhu. New digital signature scheme attaining immunity to adaptive-chosen message attack. *Chinese Journal of Electronics*, 10(4):484–486, October 2001.

[104] Huafei Zhu. A formal proof of Zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155, 2003. `http://eprint.iacr.org/`.

Sven Schäge

Contact Information

sven.schaege@rub.de

Personal Data

Citizenship: Germany

Education

Doctoral Degree (Dr.-Ing.) in Electrical Engineering, Ruhr-Universität Bochum **06/2011**
- Pass with distinction
- Thesis: Efficient and Provably Secure Signature Schemes in the Standard Model
- Advisor: Professor Dr. Jörg Schwenk
- Reader: Professor Dr. Eike Kiltz
- Field of research: provable security, public key cryptography, digital signature schemes in the standard model

Diploma Degree (Dipl.-Ing.) in IT-Security **12/2006**
- Final average grade: 1.3
- Thesis: Efficient Hash Collision Search Strategies on Special-Purpose Hardware
- Advisor: Professor Dr.-Ing. Christof Paar
- Field of research: embedded security

Student of IT-Security at Ruhr-University Bochum **10/2001-12/2006**

Rettungssanitäter ('advanced' emergency medical technician) Examination, Wiesbaden **2/2001**

Community Service at Arbeiter-Samariter-Bund, Ortsverein Bochum **10/2000-08/2001**
- Worked in the emergency medical service of Bochum as Rettungshelfer (emergency medical technician).

Albert-Einstein-Schule Bochum (Grammar School) **08/1991-06/2000**
- Attended bilingual (English-German) branch from 5th to 10th class.
- History, social studies, geography were taught in English.
- Abitur (A-level): final average grade 1.6

Recognition

Ruhr-Universität Bochum **since 06/2008**

- Fellow of the Research-School

Ruhr-Universität Bochum **09/2003**

- Award for best AES implementation in class

German Physical Society **06/2000**

- Award of German Physical Society for outstanding grades in physics

Experience

Department of Computer Science **since 06/2011**
University College London

Research Associate (PostDoc)

- UCL in global university rankings:
  QS, THE, ARWU: 4,4,21 (2009); 4,22,21 (2010); 7,17,20 (2011).
- Research interests:
  signature schemes, encryption schemes, zero-knowledge proofs.

International School of IT Security AG **04/2009-04/2011**
Bochum, Germany

Correspondence Course Supervisor for Network Security

- Supervised students in the course Network Security.

Chair for Network and Data Security **02/2007-06/2011**
Dept. of Electr. Eng. and Information Sciences, Ruhr-Universität Bochum

Research Associate

- Supervised, instructed and graded students in several courses, seminars,. . . .

Dept. of Electr. Eng. and Inform. Sciences **02/2007-04/2011**
Ruhr-Universität Bochum

Department's Academic Advisor

- Advised and supported IT-Security students.
- Offered career counseling.
- Organized student exchange programs.
- Represented the department at career events.

Chair for Embedded Security **12/2006-02/2007**
Dept. of Electr. Eng. and Information Sciences, Ruhr-Universität Bochum

Security Consultant

- Worked as an external security consultant in an industry project on Pay-TV security.

G DATA Software AG **03/2006**
Bochum, Germany

Teaching

- Organized and held a professional education event for technical employees. Topics: basics of IT-security, malware, firewalls, spam.

escrypt GmbH **01/2006-05/2006**
Bochum, Germany

Security Consultant

- Worked as an external security consultant in an industry project.

G DATA Software AG **04/2005-01/2007**
Bochum, Germany

Technical Support

- Provided technical support for G DATA's security products (e.g. antivirus software, firewall software).

Chair for System Security **07/2004-01/2005**
Dept. of Electr. Eng. and Inform. Sciences, Ruhr-Universität Bochum

Student Research Assistant

- Developed solutions for trusted computing applications based on XrML (eXtensible rights Markup Language).

Chair for Mathematics and Computer Science **10/2003-04/2004**
Department of Mathematics, Ruhr-Universität Bochum

Student Research Assistant

- Assisted Discrete Mathematics instructional team.
- Advised and instructed students. Graded exercises.

Genius Bytes Software Solutions GmbH **09/2003-06/2004**
Bochum, Germany

Software Developer

- Developed XML and smart card based applications.

Genius Bytes Software Solutions GmbH **06/2003-09/2003**
Bochum, Germany

13 Weeks Internship

- Developed smart card based applications.

Dept. of Electr. Eng. and Inform. Sciences **04/2003-08/2003**
Ruhr-Universität Bochum

Tutor

- Advised and trained first semester students as a tutor of the department.
- Lectured a weekly course on JAVA.

Volunteering in Emergency Medical Service **09/2001-07/2003**
Arbeiter-Samariter-Bund Bochum, Germany

Emergency Medical Technician

- Worked in the emergency medical service in Bochum.

Publications

Books

Finding Hash Collisions **2008**

- Sven Schäge. Finding Hash Collisions. VDM Verlag Dr. Müller, ISBN: 3639066135.

Journals

Security of MS Cardspace and related Single-Sign-On Protocols **2008**

- Xuan Chen, Christoph Löhr, Sebastian Gajek, Sven Schäge. Die Sicherheit von MS Cardspace und verwandten Single-Sign-On Protokollen. Datenschutz und Datensicherheit - DuD. Volume 32, Number 8, 515–519. Vieweg Verlag, August 2008.

Refereed Conference Proceedings

Tight Proofs for Signature Schemes without Random Oracles **2011**

- Sven Schäge. Accepted for EUROCRYPT 2011, Tallinn, May 15th-19th, 2011. Springer, LNCS 6632, 2011.

Generic Compilers for Authenticated Key Exchange **2010**

- Tibor Jager, Florian Kohler, Sven Schäge, Jörg Schwenk. ASIACRYPT 2010, Singapore, December 5th-9th, 2010. Springer, LNCS 6477, 2010.

Towards an Anonymous Access Control and Accountability Scheme for Cloud Computing **2010**

- Meiko Jensen, Sven Schäge, Jörg Schwenk. 3rd International Conference on Cloud Computing (CLOUD), Miami, Florida, USA, July 5th-10th 2010. IEEE Computer Society, 2010.

A New RSA-Based Signature Scheme **2010**

- Sven Schäge, Jörg Schwenk. AFRICACRYPT 2010, Stellenbosch, South Africa, May 3th-6th, 2010. Springer, LNCS 6055, 2010.

A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys **2010**

- Sven Schäge, Jörg Schwenk. Financial Cryptography Fourteenth International Conference, FC 2010, Tenerife, Spain, January 25th-28th, 2010. Springer, LNCS 6052, 2010.

Twin Signatures, Revisited **2009**

- Sven Schäge. Provable Security Third International Conference, ProvSec 2009, Guangzhou, China, November 11th-13th, 2009. Springer, LNCS 5848, 2009.

Code Voting with Linkable Group Signatures **2008**

- Jörg Helbach, Sven Schäge, Jörg Schwenk. Code Voting with Linkable Group Signatures. 3rd International Conference, Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC, in Castle Hofen, Bregenz, Austria, August 6th-9th, 2008. In GI, LNI 131, 2008.

Efficient Hash Collision Search Strategies on Special-Purpose Hardware **2007**

- Tim Güneysu, Christof Paar, Sven Schäge. Efficient Hash Collision Search Strategies on Special-Purpose Hardware. Western European Workshop on Research in Cryptology, WeWORC Workshop 2007, July 4th-6th, 2007, Bochum, Germany. Springer, LNCS 4945, 2007.

Technical Reports

A Standard-Model Security Analysis of TLS **05/2011**

- Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. IACR Cryptology ePrint Archive, Report 2011/219. Available online at `http://eprint.iacr.org/`.

Invited Talks

Scientific Talks

14. Kryptotag of German Gesellschaft für Informatik e.V. held at Ruhr-Universität Bochum **03/2011**

- Yearly scientific event organized by the group 'Angewandte Kryptologie' (applied cryptology) of the German Gesellschaft für Informatik e.V. . The 14th Kryptotag was held at Ruhr-Universität Bochum. I was invited to give a talk on tightly secure signature schemes.

Cryptography Seminar of Institute of Mathematical Research of Rennes, Université de Rennes 1 **12/2011**

- Regular seminar of the Institute of Mathematical Research of Rennes with international speakers. I was invited to give a talk on tightly secure signature schemes.

Professional Activities

Workshop/Conference Organization

IT-Security Career Event **06/2008, 06/2009, 06/2010**
Bochum, Germany

- Co-organized a career event that focussed on students of IT-Security. About 20 industrial companies attended, for example BMW, T-Systems, WestLB, Deutsche Bank, IBM, KPMG, Ernst & Young, McKinsey, Deutsche Bank, Boston Consulting Group, ...

Organization ECC 2004 **09/2004**
Bochum, Germany

- Helped to organize the 8th workshop on Elliptic Curve Cryptography (ECC 2004).

Reviews/Sub-Reviews

- The 15th IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2012), to be held in Darmstadt, Germany, May 21-23 2012.
- Th 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012), to be held in Cambridge, UK, April 15-19 2012.
- The 17th International Conference on Theory and Apllication of Cryptology and information Security (ASIACRYPT '11), to be held in Seoul, Korea, December 4-8 2011.
- Theoretical Computer Science, Elsevier 2011.
- 9th International Conference on Applied Cryptography and Network Security (ACNS '11), held in Nerja (Malaga), Spain, June 7-10 2011.
- The 7th Information Security Practice and Experience Conference (ISPEC 2011), held in Guangzhou, China, May 30 - June 1 2011.
- Journal of Systems and Software, Volume 84, Elsevier 2011.
- The 14th IACR International Conference on Practice and Theory of Public Key Cryptography (PKC 2011), held in Taormina, Italy, March 6-9 2011.
- The 15th European Symposium on Research in Computer Security (ESORICS 2010), held in Athens, Greece, September 20-22 2010.
- 3rd IEEE Intl. Symposium on Security in Networks and Distributed Systems (SSNDS) 2007, held at Niagara Falls, Canada, May 21-23 2007.
- The 10th International Conference on Information Security and Cryptology (ICISC 2007) held in Seoul, Korea, November 29-30 2007.
- Workshop on Cryptographic Hardware and Embedded Systems 2006 (CHES 2006) held in Yokohama, Japan, October 10-13 2006. Top conference in the field.

Teaching

Network Security I **2007, 2008, 2010**

- Security analyses of various practical systems. Broadcast Encryption, Pay-TV, DVD Encryption, Mobile Telephony (GSM, UMTS), WLAN (IEEE 802.11), Firewalls, Intrusion Detection Systems, Malware, Web Services (XML Security, Microsoft Passport, WS-Security).
- Gave guest lectures, held exercise courses and set questions in the final exam.

Network Security II **2007, 2008, 2009, 2010**

- Security analyses of various practical systems. OpenPGP, S/MIME, SSL, DNSSEC, VPN (IPSec, PPTP, IP Multicast).
- Gave guest lectures, held exercise courses and set questions in the final exam.

Cryptographic Protocols **2008, 2009**

- Topics: Provable Security, Commitment Schemes, Zero-Knowledge-Protocols, Multiparty Computations (Shared Secret Schemes, Electronic Poker, Secure Circuit Evaluation), Universal Composability.
- Gave guest lectures, held exercise courses and set questions in the final exam.

Seminars, Practical Courses, Projects **2007, 2008, 2009, 2010**

- Supervised students.
- Developed experiment descriptions.
- Organized seminars.

Industry Projects

Gematik GmbH Research Project **09/2009-08/2010**

- Worked on Germany's prospective electronic health insurance card.