J. Cryptol. (2015) 28: 641–670
DOI: 10.1007/s00145-013-9173-6

Journal of CRYPTOLOGY

# Tight Security for Signature Schemes Without Random Oracles*

Sven Schäge†
University College London, London, UK
s.schage@ucl.ac.uk

Communicated by Tal Rabin

Received 31 July 2012
Online publication 21 December 2013

**Abstract.** We present the first tight security proofs for two general classes of Strong RSA (SRSA) based signature schemes. Among the covered signature schemes are the signature schemes by Cramer–Shoup, Zhu, Fischlin, and the SRSA-based Camenisch–Lysyanskaya scheme with slightly modified parameter sizes. We also present two variants of our signature classes in bilinear groups that output very short signatures. Similarly to before, these variants have tight security proofs under the Strong Diffie–Hellman (SDH) assumption. We so obtain very efficient SDH-based variants of the Cramer–Shoup, Fischlin, and Zhu signature scheme and the first tight security proof for the recent Camenisch–Lysyanskaya scheme that was proposed and proven secure under the SDH assumption. Central to our results is a new proof technique that allows the simulator to avoid guessing which of the attacker's signature queries will be reused in the forgery. In contrast to previous proofs, our security reduction does not lose a factor of $q$ here, where $q$ is the number of signature queries made by the adversary.

**Key words.** Signature scheme, Signature class, Tight security, SRSA, SDH, Standard model.

## 1. Introduction

*Provable Security and Tight Reductions* The central idea of provable security is to design a cryptographic scheme in such a way that if an attacker $\mathcal{A}$ could efficiently break its security properties then one can also construct an efficient algorithm $\mathcal{B}$, to break a supposedly hard problem. In this way, we prove the security of the scheme *by reduction* to the hardness assumption. Now, if $\mathcal{B}$ has almost the same success probability as $\mathcal{A}$ while running in roughly the same time we say that the security reduction is tight. Otherwise, the security reduction is said to be loose. Of course, any scheme is trivially secure under the sole assumption that it is hard to break. However, cryptographers are

* This paper is an extended version of [21].
† S. Schäge was supported by EPSRC Grant EP/J009520/1.

particularly interested in security reductions where a (complex) scheme whose security is defined via an *interactive* security game is shown to be secure under *non-interactive* security assumptions only (e.g. under the well-known factoring assumption). This is because non-interactive security assumptions can be analyzed much better than interactive ones.

*Motivation* Cryptographers are interested in tight security proofs as they allow for shorter security parameters and better efficiency. This paper was also motivated by the observation that for several of the existing Strong RSA (SRSA) 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). For example, in some schemes the signing algorithms injectively map messages to primes. Since prime generation depends on the input message, these schemes cannot pre-compute primes in times of relative low workload. As prime generation is one of the most costly operations in these schemes, their reaction time is considerably lower than in comparable schemes that allow to pre-compute primes. In 2007, Chevallier-Mames and Joye addressed this problem in the following way [7]: they took a tightly secure signature scheme, the Gennaro–Halevi–Rabin scheme [11], and improved its efficiency by re-designing one of its most time-consuming functions. 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 [11] by *choosing* a random prime and making use of a chameleon hash function to map the input message to that prime. In this way, primes can be pre-computed and the reaction time of the signer can significantly be lowered. Unfortunately, their approach also has a serious disadvantage as the so computed primes are much larger than in the original GHR scheme. In addition to that, there is also a general disadvantage when improving a scheme's efficiency (or that of its security reduction) by *modifying* its algorithms: the resulting efficiency benefits only apply to *new* implementations. Existing implementations (of the original scheme) are not affected. In contrast, as security reductions are merely thought experiments (that do not actually have to be implemented), improving the tightness of a scheme's security reduction benefits both existing and new implementations. In this paper, we therefore 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 [2]. In contrast to Bernstein, we concentrate on schemes that are secure in the standard model.

*Contribution* In this work, we address the following question: are there tight security proofs for the existing, practical signature schemes by Cramer–Shoup [9], Zhu [24], Camenisch–Lysyanskaya [4] and Fischlin [10] (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 (with slightly modified parameter lengths), 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'. Signatures always consist of three components, a random value $r$, a random prime $e$, and an integer $s$ such that

$$s = \left(u v^r w^{z(r,m)}\right)^{\frac{1}{e}} \bmod n$$

for some RSA modulus $n$ and some public generators $u, v, w$. The concrete schemes only differ in how they instantiate the combining function $z(r, m)$. In a similar way, we introduce a second general class of signature schemes called 'chameleon hash scheme' that is inspired by the structure of the Cramer–Shoup signature scheme. Here, signatures always look like

$$s = \left(u v^{ch(r,m)}\right)^{\frac{1}{e}} \bmod n$$

where $ch(r, m)$ refers to a concrete instantiation of a chameleon hash function. 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. Unfortunately, the security proof of the SRSA-based chameleon hash scheme does not directly transfer to the original Cramer–Shoup signature scheme. This is simply because in the Cramer–Shoup scheme the key material of the chameleon hash function is not chosen independently from the rest of the signature scheme. Nevertheless, the proof of the Cramer–Shoup signature scheme is technically very similar to the proof of the chameleon hash scheme (and we also provide a separate proof of tight security of the Cramer–Shoup signature scheme). Finally, we show that our results 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. The first conclusion is particularly surprising and rather exceptional; usually new progress in cryptanalysis makes deployed cryptographic systems 'less secure' than expected.

*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 random 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.

On a more technical level, we first exploit that two of the components of the signatures $r$ and $e$ are drawn uniformly at random. As a consequence, the simulator can compute the entire set $R$ of randomness pairs $\{(r_i, e_i)\}_{i=1,\ldots,q}$ that it is going to use in the responses to the adversary's signature queries *before* setting up the public key. Given $R$, the simulator can now carefully design a linear function $f$ with several useful properties. First, since it is linear, $f$ can be embedded in the exponents of only two (random-looking) public group generators, for example in $u$ and $v$ in the chameleon hash scheme. Second, the properties of $f$ guarantee that the simulator can successfully simulate the signing oracle (if it actually uses the randomness pairs in $R$ to respond to the signature queries). Third, if the adversary computes a forgery $r^*, e^*, s^*$ where only a single random value has been used in a previous signature (i.e. either $e^* = e_i$ or $r^* = r_i$ for some $1 \leq i \leq q$ but not both), the simulator can break the underlying security assumption.

*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. We subsequently give a brief overview on the available results, also see Table 1. In 1988, Goldwasser, Micali and Rivest published the first provably secure, but inefficient signature scheme [12]. More than a decade later, in 1999, Gennaro, Halevi, and Rabin [11] presented a signature scheme that is secure in the standard model under the Flexible or Strong RSA assumption (SRSA). This scheme is more efficient, both the key and the signature size are less than two group elements (à 1024 bits), but as a drawback, it relies on an impractical function that injectively maps messages to primes [8,17]. Advantageously, the Gennaro–Halevi–Rabin signature scheme is known to have a tight security proof. At the same time and also based on the SRSA assumption, Cramer and Shoup [9] proposed an efficient standard model signature scheme, that unlike [11] does not require to map messages to primes. In contrast, primes can be drawn uniformly at random from the set of primes of a given bitlength. Based on this work, Zhu [23,24], Fischlin [10], Camenisch and Lysyanskaya [4], and Hofheinz and Kiltz [13] in the following years presented further SRSA-based schemes. These schemes are either more efficient than the Cramer–Shoup scheme or very suitable in protocols for issuing signatures on committed values. In 2011, Joye presented a revised version of the Zhu scheme as, for the parameter sizes given in [23,24], the original scheme suffers from a flaw in the security proof [14]. Joye's version uses slightly different parameter lengths what fixes the main problem in the security proof. In 2011, Catalano, Fiore, and Warinschi, presented an adaptive definition of pseudo-free groups, where the adversary is also allowed to learn solutions to other non-trivial equations before having to solve his challenge equation [6]. The authors then showed that (1) adaptive pseudo-free groups can generically be used to construct digital signature schemes (and even network coding signatures) and that (2) the well-known RSA group is an adaptive pseudo-free group under the SRSA assumption. In this way, they can generalize most of the existing SRSA-based signature

**Table 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.

| Signature scheme | Security assumption | Security loss | | Prime generation |
|---|---|---|---|---|
| | | Original reduction | Our reduction | |
| Gennaro–Halevi–Rabin [11] | SRSA | O(1) | | **INJ** |
| Cramer–Shoup [9] | SRSA | O(q) | O(1) | |
| Naccache–Pointcheval–Stern [17] | SRSA | O(1) | | **INJ** |
| Fischlin [10] | SRSA | O(q) | O(1) | |
| Zhu [23,24] (Joye [14]) | SRSA | O(q) | O(1) | |
| Camenisch–Lysyanskaya [4]* | SRSA | O(q) | O(1) | |
| Chevallier-Mames–Joye [7] | SRSA | O(1) | | |
| Hofheinz–Kiltz [13] | SRSA | O(q) | | |
| Boneh–Boyen [3] | SDH | O(1) | | |
| Camenisch–Lysyanskaya [1,5,18] | SDH | O(q) | O(1) | |
| Hofheinz–Kiltz [13] | SDH | O(q) | | |

* Indicates that we use modified parameter sizes (as compared to the original work).

schemes in a single framework. Unfortunately, their work does not consider concrete security bounds. As a consequence, it does not cover highly optimized schemes like the SRSA-based signature scheme by Hofheinz and Kiltz [13]. In 2004, Boneh and Boyen presented the first hash-and-sign signature scheme that makes use of bilinear groups [3]. The big advantage of bilinear groups is the very compact representation of group elements. The Boneh–Boyen signature scheme is proven tightly 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 [5]. Unlike the Boneh–Boyen scheme, their scheme is proven secure under the LRSW assumption that was proposed by Lysyanskaya, Rivest, Sahai, and Wolf [16]. 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 four years later in [1,18]. Similar to the SRSA-based Camenisch–Lysyanskaya scheme the security proof of the SDH scheme is loose.

## 2. Preliminaries

Before presenting our results we briefly review the necessary formal definitions. For convenience, we also describe two general key generation procedures (settings) in Sects. 2.7 and 2.8. When describing our signature schemes in Sect. 3 we will refer to the corresponding setting and only describe the signature generation and verification algorithms.

### 2.1. *Notation*

For $a, b \in \mathbb{Z},\ a \leq b$ we write $[a; b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$. For a string $x$, we write $|x|_2$ to denote its bit length. If $z \in \mathbb{Z}$, we write $|z|$ to denote the

absolute value of $z$. For a set $X$, we use $|X|$ to refer to its size and $x \xleftarrow{\$} X$ to indicate that $x$ is drawn from $X$ uniformly at random. For $n \in \mathbb{N}$, we use $QR_n$ to denote the set of quadratic residues modulo $n$, i.e. $QR_n = \{x | \exists y \in \mathbb{Z}_n^* : y^2 = x \bmod n\}$. If $\mathcal{A}$ is an algorithm we write $\mathcal{A}(x_1, x_2, \ldots)$ to denote that $\mathcal{A}$ has input parameters $x_1, x_2, \ldots$. Accordingly, $y \leftarrow \mathcal{A}(x_1, x_2, \ldots)$ means that $\mathcal{A}$ outputs $y$ when running with inputs $x_1, x_2, \ldots$. We write PPT (probabilistic polynomial time) for randomized algorithms that run in polynomial time. We write $\kappa \in \mathbb{N}$ to indicate the security parameter and $1^\kappa$ to describe the string that consist of $\kappa$ ones.

### 2.2. *Signature Scheme*

A digital signature scheme $\mathsf{SIG} = (\mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ consists of three algorithms. The PPT algorithm $\mathsf{SIG.KeyGen}$ on input the security parameter $\kappa \in \mathbb{N}$ in unary generates a secret and public key pair $(SK, PK)$. The PPT algorithm $\mathsf{SIG.Sign}$ takes as input a secret key $SK$ and the message $m$ in the message space $\mathcal{M}_{\mathsf{SIG}}$ and outputs a signature $\sigma$ in the signature space $\mathcal{S}_{\mathsf{SIG}}$. Finally, the deterministic polynomial time algorithm $\mathsf{SIG.Verify}$ processes a public key $PK$, a message $m$ and a signature $\sigma$ to check whether $\sigma$ is a legitimate signature on $m$ signed by the holder of the secret key corresponding to $PK$. Accordingly, the algorithm outputs 1 to indicate a successful verification and 0 otherwise.

### 2.3. *Strong Existential Unforgeability*

The standard notion of security for signature schemes is due to Goldwasser, Micali and Rivest [12]. We use a slightly stronger definition called strong existential unforgeability. The signature scheme $\mathsf{SIG} = (\mathsf{SIG.KeyGen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$ is strongly existentially unforgeable under an adaptive chosen message attack if it is infeasible for a forger, who only knows the public key and the global parameters, to produce, after obtaining polynomially (in the security parameter) many signatures $\sigma_1, \ldots, \sigma_q$ on messages $m_1, \ldots, m_q \in \mathcal{M}_{\mathsf{SIG}}$ of its choice from a signing oracle $\mathcal{O}(SK, \cdot)$, a new message/signature pair.

**Definition 1.** We say that $\mathsf{SIG}$ is $(q, t, \epsilon)$-secure, if for all $t$-time adversaries $\mathcal{A}$ that send at most $q$ queries to the signing oracle $\mathcal{O}(SK, \cdot)$ it holds that

$$\Pr\left[(SK, PK) \leftarrow \mathsf{SIG.KeyGen}(1^\kappa),\ \left(m^*, \sigma^*\right) \leftarrow \mathcal{A}^{\mathcal{O}(SK,\cdot)}(PK),\right.$$
$$\left.\mathsf{SIG.Verify}\left(PK, m^*, \sigma^*\right) = 1\right] \leq \epsilon,$$

where the probability is over the random coins of $\mathsf{SIG.KeyGen}$ and $\mathcal{A}$ and $(m^*, \sigma^*) \in \mathcal{M}_{\mathsf{SIG}} \times \mathcal{S}_{\mathsf{SIG}}$ is not among the message/signature pairs obtained using $\mathcal{O}(SK, \cdot)$, i.e. $((m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\})$.

### 2.4. *Collision-Resistant Hashing*

A hash function $\mathsf{H} = (\mathsf{H.KeyGen}, \mathsf{H.Eval})$ consist of two algorithms. The PPT $\mathsf{H.KeyGen}$ takes as input $1^\kappa$ and outputs a hash key $k_{\mathsf{H}}$. The deterministic algorithm $\mathsf{H.Eval}$ takes as input the hash key $k_{\mathsf{H}}$ and a message $m \in \mathcal{M}_{\mathsf{H}}$ and outputs a hash value $z \in \mathcal{H}_{\mathsf{H}}$.

**Definition 2** (Collision-Resistant Hash Function). $\mathsf{H}$ is called $(t_{\mathsf{H}}, \epsilon_{\mathsf{H}})$-collision-resistant if for all $t_{\mathsf{H}}$-time adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{c} k_{\mathsf{H}} \leftarrow \mathsf{H.KeyGen}(1^\kappa),\ (m, m') \leftarrow \mathcal{A}(k_{\mathsf{H}}),\ m \neq m', \\ m, m' \in \mathcal{M}_{\mathsf{H}},\ \mathsf{H.Eval}(k_{\mathsf{H}}, m) = \mathsf{H.Eval}(k_{\mathsf{H}}, m') \end{array}\right] \leq \epsilon_{\mathsf{H}} = \epsilon_{\mathsf{H}}(\kappa),$$

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

If $\mathsf{H}$ is used in a signature scheme, we assume that $k_{\mathsf{H}} \leftarrow \mathsf{H.KeyGen}(1^\kappa)$ is generated in the key generation algorithm $\mathsf{SIG.KeyGen}$ and made part of the public key $PK$. If the hash key $k_{\mathsf{H}}$ is clear from the context we write $h(m)$ as a shortcut for $\mathsf{H.Eval}(k_{\mathsf{H}}, m)$.

### 2.5. *Chameleon Hash Function*

A useful tool in many of the signature schemes without random oracles is a chameleon hash function [15]. A chameleon hash function $\mathsf{CH} = (\mathsf{CH.KeyGen}, \mathsf{CH.Eval}, \mathsf{CH.Coll})$ consists of three algorithms. The PPT algorithm $\mathsf{CH.KeyGen}$ takes as input the security parameter $1^\kappa$ and outputs a secret key $SK_{\mathsf{CH}}$ and a public key $PK_{\mathsf{CH}}$. Given $PK_{\mathsf{CH}}$, a random $r$ from a randomization space $\mathcal{R}_{\mathsf{CH}}$ and a message $m$ from a message space $\mathcal{M}_{\mathsf{CH}}$, the algorithm $\mathsf{CH.Eval}$ outputs a chameleon hash value $c$ in the hash space $\mathcal{C}_{\mathsf{CH}}$. Analogously, $\mathsf{CH.Coll}$ deterministically outputs, on input $SK_{\mathsf{CH}}$ and $(r, m, m') \in \mathcal{R}_{\mathsf{CH}} \times \mathcal{M}_{\mathsf{CH}} \times \mathcal{M}_{\mathsf{CH}}$, $r' \in \mathcal{R}_{\mathsf{CH}}$ such that $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, m, r) = \mathsf{CH.Eval}(PK_{\mathsf{CH}}, m', r')$.

**Definition 3** (Collision-Resistant Chameleon Hash Function). We say that $\mathsf{CH}$ is $(\epsilon_{\mathsf{CH}}, t_{\mathsf{CH}})$-collision-resistant if for all $t_{\mathsf{CH}}$-time adversaries $\mathcal{A}$ that are only given $PK_{\mathsf{CH}}$ it holds that

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

where the probability is over the random choices of $PK_{\mathsf{CH}}$ and the coin tosses of $\mathcal{A}$.

We also require that for an arbitrary but fixed public key $PK_{\mathsf{CH}}$ output by $\mathsf{CH.KeyGen}$, all messages $m \in \mathcal{M}_{\mathsf{CH}}$ generate equally distributed hash values when drawing $r \in \mathcal{R}_{\mathsf{CH}}$ uniformly at random and outputting $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, r, m)$. In the following, if $\mathsf{CH}$ is used in a signature scheme we assume that $(SK_{\mathsf{CH}}, PK_{\mathsf{CH}}) \leftarrow \mathsf{CH.KeyGen}(1^\kappa)$ is generated as part of $\mathsf{SIG.KeyGen}$ and $PK_{\mathsf{CH}}$ is made part of the public key.[1] If the keys are obvious from the context, we write $ch(r, m)$ for $\mathsf{CH.Eval}(PK_{\mathsf{CH}}, r, m)$ and $ch^{-1}(r, m, m')$ for $\mathsf{CH.Coll}(SK_{\mathsf{CH}}, r, m, m')$.

The security of chameleon hash functions can be based on well-analyzed standard assumptions like the discrete logarithm assumption [15] or the factoring assumption [15,22]. Since the factoring assumption is weaker than the SRSA assumption and the discrete logarithm assumption is weaker than the SDH assumption, we can use chameleon hash functions as a building block for SRSA and SDH-based signature schemes without relying on additional complexity assumptions.

[1] If we use the signature scheme as an online-offline signature scheme [22], $SK_{\mathsf{CH}}$ is also part of the scheme's secret key.

### 2.6. *Combining Function*

In this section, we introduce a new class of randomization functions called combining functions. A combining function is a function in two input parameters, a message $m$ (that is chosen by the adversary) and a random value $r$ that map to some output value $z$. Intuitively, security requires that it behaves computationally (or statistically) close to a bijection in both $m$ and $r$. In particular, it should be hard to (1) find $m$ such that for a randomly chosen output value $z$ there exists no $r$ mapping $m$ to $z$ and (2) compute distinct $m, m'$ that map to the same output value (for some common randomness $r$). Additionally, we want that inversion is easy, i.e. for given $m$, $z$ we can efficiently compute a suitable $r$ such that $m$ and $r$ map to $z$. We will subsequently use the concept of combining functions to generalize several existing signature schemes. Let us begin with the formal definition.

A combining function $\mathsf{CMB} = (\mathsf{CMB.KeyGen}, \mathsf{CMB.Eval})$ consist of two algorithms. On input $1^\kappa$ the PPT $\mathsf{CMB.KeyGen}$ outputs a (combining) key $k_{\mathsf{CMB}}$. On input $k_{\mathsf{CMB}}$, randomness $r \in \mathcal{R}_{\mathsf{CMB}}$ and message $m \in \mathcal{M}_{\mathsf{CMB}}$, the deterministic algorithm $\mathsf{CMB.Eval}$ outputs a value $z \in \mathcal{Z}_{\mathsf{CMB}}$.

**Definition 4** (Secure Combining Functions). We say that $\mathsf{CMB}$ is $(t_{\mathsf{CMB}}, \epsilon_{\mathsf{CMB}}, \delta_{\mathsf{CMB}})$-combining if there exist functions $\epsilon_{\mathsf{CMB}}(\kappa)$ and $\delta_{\mathsf{CMB}}(\kappa)$ and for $k_{\mathsf{CMB}} \leftarrow \mathsf{CMB.KeyGen}(1^\kappa)$ the following properties hold:

1. For all $m \in \mathcal{M}_{\mathsf{CMB}}$ it holds that $|\mathcal{R}_{\mathsf{CMB}}| = |\mathcal{Z}_m|$ where $\mathcal{Z}_m \subseteq \mathcal{Z}_{\mathsf{CMB}}$ is defined as $\mathcal{Z}_m = \mathsf{CMB.Eval}(k_{\mathsf{CMB}}, \mathcal{R}_{\mathsf{CMB}}, m)$ (i.e. $\mathsf{CMB.Eval}$ is injective in the input parameter $r$ for all $m$).
2. There exists an efficient deterministic algorithm $\mathsf{CMB.Invert}$ that on input $k_{\mathsf{CMB}}$, $t \in \mathcal{Z}_{\mathsf{CMB}}$, and $m \in \mathcal{M}_{\mathsf{CMB}}$ outputs $\perp$ if $t \notin \mathcal{Z}_m$ and $r$ with $t = \mathsf{CMB.Eval}(k_{\mathsf{CMB}}, r, m)$ otherwise.
3. The uniform distribution on $\mathcal{Z}_m$ and $\mathcal{Z}_{\mathsf{CMB}}$ are $\delta_{\mathsf{CMB}}$-indistinguishable, i.e. for all $m \in \mathcal{M}_{\mathsf{CMB}}$, $t \xleftarrow{\$} \mathcal{Z}_{\mathsf{CMB}}$, and $r' \xleftarrow{\$} \mathcal{R}_{\mathsf{CMB}}$ it holds for all $t_{\mathsf{CMB}}$-time attackers $\mathcal{A}$ that
$$\left|\Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, \mathsf{CMB.Invert}(k_{\mathsf{CMB}}, t, m)\right) = 1\right]\right| \leq \delta_{\mathsf{CMB}},$$
where the probability is taken over the random bits of $\mathsf{CMB.KeyGen}(1^\kappa)$ and $\mathcal{A}$ and over the random choices of $t$ and $r'$.
4. For all $r \in \mathcal{R}_{\mathsf{CMB}}$, it holds for all $t_{\mathsf{CMB}}$-time attackers $\mathcal{A}$ that
$$\Pr\left[\begin{array}{c}(m, m') \leftarrow \mathcal{A}(k_{\mathsf{CMB}}, r),\ m, m' \in \mathcal{M}_{\mathsf{CMB}}, \\ m \neq m',\ \mathsf{CMB.Eval}(k_{\mathsf{CMB}}, r, m) = \mathsf{CMB.Eval}(k_{\mathsf{CMB}}, r, m')\end{array}\right] \leq \epsilon_{\mathsf{CMB}},$$
where the probability is taken over the random bits of $\mathcal{A}$ and $\mathsf{CMB.KeyGen}(1^\kappa)$.

In the following, we assume that when used in signature schemes, $k_{\mathsf{CMB}} \leftarrow \mathsf{CMB.KeyGen}(1^\kappa)$ is generated during the key generation phase of the signature scheme. If the key $k_{\mathsf{CMB}}$ is clear from the context we write $z(r, m)$ short for $\mathsf{CMB.Eval}(k_{\mathsf{CMB}}, r, m)$ and $z^{-1}(t, m)$ short for $\mathsf{CMB.Invert}(k_{\mathsf{CMB}}, t, m)$. If security is independent from a random choice of the key, we may, for consistency, assume that $\mathsf{CMB.KeyGen}$ just outputs a special, fixed key $\top$.

**Table 2.** Examples of statistically secure combining functions. For all $\mathsf{CMB}_i$ $i = 1, 2, 3$ $\mathsf{CMB}_i.\mathsf{KeyGen}(1^\kappa)$ just outputs the special key $\top$. Let $l, l_r, l_m \in \mathbb{N}$, $l_r > l_m$ and $p$ be prime.

| | $z(r, m)$ | $\mathcal{R}_{\mathsf{CMB}}$ | $\mathcal{M}_{\mathsf{CMB}}$ | $\mathcal{Z}_{\mathsf{CMB}}$ | Combining |
|---|---|---|---|---|---|
| $\mathsf{CMB}_1$ | $r + m \bmod p$ | $\mathbb{Z}_p$ | $\mathbb{Z}_p$ | $\mathbb{Z}_p$ | $(\cdot, 0, 0)$ |
| $\mathsf{CMB}_2$ | $r \oplus m$ | $\{0,1\}^l$ | $\{0,1\}^l$ | $\{0,1\}^l$ | $(\cdot, 0, 0)$ |
| $\mathsf{CMB}_3$ | $r + m$ | $[0; 2^{l_r} - 1]$ | $[0; 2^{l_m} - 1]$ | $[0; 2^{l_r} + 2^{l_m} - 2]$ | $(\cdot, 0, 2^{l_m - l_r})$ |

In Table 2, we present three concrete examples ($\mathsf{CMB}_1$, $\mathsf{CMB}_2$, $\mathsf{CMB}_3$) of *statistically* secure combining functions. The following lemma shows that these examples are indeed combining functions with respect to Definition 4.

**Lemma 1.** $\mathsf{CMB}_1$ *and* $\mathsf{CMB}_2$ *are* $(\cdot, 0, 0)$*-combining.* $\mathsf{CMB}_3$ *is* $(\cdot, 0, 2^{l_m - l_r + 1})$*-combining.*

**Proof.** Let us first analyze $\mathsf{CMB}_1$ and $\mathsf{CMB}_2$. We have that $\mathcal{M}_{\mathsf{CMB}} = \mathcal{R}_{\mathsf{CMB}} = \mathcal{Z}_{\mathsf{CMB}} = \mathcal{Z}_m$ for all $m \in \mathcal{M}_{\mathsf{CMB}}$ and we can efficiently compute $r$ as $r = t - m \bmod p$ or $r = t \oplus m$ for *all* given $t \in \mathcal{Z}_{\mathsf{CMB}}$ and $m \in \mathcal{M}_{\mathsf{CMB}}$. Furthermore, since $z$ is bijective in both input parameters $z^{-1}(t, m)$ is uniformly distributed in $\mathcal{R}_{\mathsf{CMB}}$ for all $m \in \mathcal{M}_{\mathsf{CMB}}$ and random $t \in \mathcal{Z}_{\mathsf{CMB}}$. Thus, $\delta_{\mathsf{CMB}} = 0$. Finally, since $z$ is a bijection in the second input parameter, it is collision-free in both combining functions and we have that $\epsilon_{\mathsf{CMB}} = 0$. Now, let us analyze $\mathsf{CMB}_3$. For given $m \in \mathcal{M}_{\mathsf{CMB}}$ and random $t \in \mathcal{Z}_{\mathsf{CMB}}$, $z^{-1}(t, m)$ outputs $r = t - m$ if $t - m \in \mathcal{R}_{\mathsf{CMB}}$ (i.e. if $t \in \mathcal{Z}_m$) and $\perp$ otherwise. To show that $z(r, \cdot)$ is collision-free, observe that $m \neq m'$ implies $r + m \neq r + m'$ for all $r \in \mathcal{R}_{\mathsf{CMB}}$. To prove the bound on $\delta_{\mathsf{CMB}}$ note that for $t' \xleftarrow{\$} \mathcal{Z}_m$, $z^{-1}(t', m)$ is uniform in $\mathcal{R}_{\mathsf{CMB}}$ since $|\mathcal{Z}_m| = |\mathcal{R}_{\mathsf{CMB}}|$ implies that $z^{-1}(\cdot, m)$ defines a bijection from $\mathcal{Z}_m$ to $\mathcal{R}_{\mathsf{CMB}}$. For $t' \xleftarrow{\$} \mathcal{Z}_m$ and $t \xleftarrow{\$} \mathcal{Z}_{\mathsf{CMB}}$ we get

$$
\begin{aligned}
&\max_{m \in \mathcal{M}_{\mathsf{CMB}}} \left| \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, \mathsf{CMB.Invert}(k_{\mathsf{CMB}}, t, m)\right) = 1\right] \right| \\
&= \max_{m \in \mathcal{M}_{\mathsf{CMB}}} \left| \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, \mathsf{CMB.Invert}\left(k_{\mathsf{CMB}}, t', m\right)\right) = 1\right] \right. \\
&\quad \left. - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, \mathsf{CMB.Invert}(k_{\mathsf{CMB}}, t, m)\right) = 1\right] \right| \\
&\leq \max_{m \in \mathcal{M}_{\mathsf{CMB}}} \Pr[t \notin \mathcal{Z}_m] = \max_{m \in \mathcal{M}_{\mathsf{CMB}}} \frac{|\mathcal{Z}_{\mathsf{CMB}}| - |\mathcal{Z}_m|}{|\mathcal{Z}_{\mathsf{CMB}}|} \\
&= \frac{|\mathcal{Z}_{\mathsf{CMB}}| - |\mathcal{R}_{\mathsf{CMB}}|}{|\mathcal{Z}_{\mathsf{CMB}}|} = \frac{2^{l_m}}{2^{l_m} + 2^{l_r}} \leq 2^{l_m - l_r}.
\end{aligned}
$$

□

Three further examples of combining functions can be obtained when first applying a $(t_{\mathsf{H}}, \epsilon_{\mathsf{H}})$-collision-resistant hash function that maps (long) messages from $\mathcal{M}_{\mathsf{H}}$ (for example $\mathcal{M}_{\mathsf{H}} = \{0,1\}^*$) to $\mathcal{H}_{\mathsf{H}} = \mathcal{M}_{\mathsf{CMB}}$. The next lemma guarantees that the results are still combining according to Definition 4.

**Lemma 2.** *Let* $\mathsf{CMB}$ *be a* $(t_{\mathsf{CMB}}, \epsilon_{\mathsf{CMB}}, \delta_{\mathsf{CMB}})$*-combining function with message space* $\mathcal{M}_{\mathsf{CMB}}$*. Let* $\mathsf{H}$ *be a* $(t_{\mathsf{H}}, \epsilon_{\mathsf{H}})$*-collision-resistant hash function with message space* $\mathcal{M}_{\mathsf{H}}$

*and hash space* $\mathcal{H}_{\mathsf{H}} = \mathcal{M}_{\mathsf{CMB}}$. *Then it holds that* $\mathsf{CMB}' = (\mathsf{CMB.KeyGen}', \mathsf{CMB.Eval}', \mathsf{CMB.Invert}')$ *is* $(\min\{t_{\mathsf{CMB}}, t_{\mathsf{H}}\}, \epsilon_{\mathsf{CMB}} + \epsilon_{\mathsf{H}}, \delta_{\mathsf{CMB}})$*-combining with message space* $\mathcal{M}_{\mathsf{CMB}'} = \mathcal{M}_{\mathsf{H}}$ *where the algorithms of* $\mathsf{CMB}'$ *are defined as follows*:

- $\mathsf{CMB.KeyGen}'(1^\kappa)$ *outputs* $k'_{\mathsf{CMB}} = (k_{\mathsf{CMB}}, k_{\mathsf{H}})$ *where* $k_{\mathsf{CMB}} \leftarrow \mathsf{CMB.KeyGen}(1^\kappa)$ *and* $k_{\mathsf{H}} \leftarrow \mathsf{H.KeyGen}(1^\kappa)$ *and*
- $\mathsf{CMB.Eval}'(k'_{\mathsf{CMB}}, r, m)$ *outputs* $z(r, h(m)) = \mathsf{CMB.Eval}(k_{\mathsf{CMB}}, r, \mathsf{H.Eval}(k_{\mathsf{H}}, m))$.

The proof of Lemma 2 is relatively straight-forward.

**Proof.** First observe that given $m \in \mathcal{M}_{\mathsf{H}}$ and $t \in \mathcal{Z}_{h(m)}$ we can always compute $r \in \mathcal{R}_{\mathsf{CMB}}$ (or $\perp$) just by finding an appropriate $r$ for $h(m)$ with $t = z(r, h(m))$ using the properties of CMB. We simply set $\mathsf{CMB.Invert}'(k'_{\mathsf{CMB}}, t, m)$ to output $\mathsf{CMB.Invert}(k_{\mathsf{CMB}}, t, \mathsf{H.Eval}(k_{\mathsf{H}}, m))$.

Now, for contradiction assume that an attacker $\mathcal{A}$ can find collisions for $\mathsf{CMB}'$ in time $\min\{t_{\mathsf{CMB}}, t_{\mathsf{H}}\}$ with probability better than $\epsilon_{\mathsf{H}} + \epsilon_{\mathsf{CMB}}$. 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 (in case $h(m) \neq h(m')$). In the first case, $\mathcal{A}$ has computed a hash collision in time equal or less $t_{\mathsf{H}}$ with a probability greater than $\epsilon_{\mathsf{H}} + \epsilon_{\mathsf{CMB}} \geq \epsilon_{\mathsf{H}}$. This contradicts the fact that H is $(t_{\mathsf{H}}, \epsilon_{\mathsf{H}})$-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 CMB in time less or equal than $t_{\mathsf{CMB}}$ with probability greater than $\epsilon_{\mathsf{H}} + \epsilon_{\mathsf{CMB}} \geq \epsilon_{\mathsf{CMB}}$. This contradicts the fact that CMB is $(t_{\mathsf{CMB}}, \epsilon_{\mathsf{CMB}}, \delta_{\mathsf{CMB}})$-combining.

To prove the probability bound $\delta_{\mathsf{CMB}}$, recall that $h(\mathcal{M}_{\mathsf{H}}) \subseteq \mathcal{M}_{\mathsf{CMB}}$. Therefore if there is an adversary $\mathcal{A}$ that can distinguish between random $r'$ and $z^{-1}(t, h(m))$ for some $m \in \mathcal{M}_{\mathsf{H}}$ and random $t \xleftarrow{\$} \mathcal{Z}_{h(m)}$ then it can also distinguish between $r'$ and $z^{-1}(t, m')$ for $m' = h(m) \in \mathcal{M}_{\mathsf{CMB}}$. In the following, all probabilities are considered over the random bits used to compute $k_{\mathsf{H}} \leftarrow \mathsf{H.KeyGen}(1^\kappa)$ and $k_{\mathsf{CMB}} \leftarrow \mathsf{CMB.KeyGen}(1^\kappa)$. We get that

$$
\begin{aligned}
\delta_{\mathsf{CMB}} &< \max_{m \in \mathcal{M}_{\mathsf{CMB}'}} \left|\Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, z^{-1}\left(t, h(k_{\mathsf{H}}, m)\right)\right) = 1\right]\right| \\
&= \max_{m \in \mathcal{M}_{\mathsf{H}}} \left|\Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, z^{-1}\left(t, h(k_{\mathsf{H}}, m)\right)\right) = 1\right]\right| \\
&= \max_{\tilde{m} \in h(k_{\mathsf{H}}, \mathcal{M}_{\mathsf{H}})} \left|\Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, z^{-1}(t, \tilde{m})\right) = 1\right]\right| \\
&\leq \max_{\tilde{m} \in \mathcal{M}_{\mathsf{CMB}}} \left|\Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, r'\right) = 1\right] - \Pr\left[\mathcal{A}\left(k_{\mathsf{CMB}}, k_{\mathsf{H}}, z^{-1}(t, \tilde{m})\right) = 1\right]\right|.
\end{aligned}
$$

So any attacker $\mathcal{A}$ against $\mathsf{CMB}'$ with success probability better than $\delta_{\mathsf{CMB}}$ is also an attacker against CMB with success probability greater or equal to $\delta_{\mathsf{CMB}}$. □

### 2.7. *The Strong RSA Setting*

**Definition 5** (Strong RSA Assumption (SRSA)). Given an RSA modulus $n = pq$, where $p, q$ are primes of size $l_n/2 - 1$ (with polynomial $l_n = l_n(\kappa)$), and an element

$u \in \mathbb{Z}_n^*$, we say that the $(t_{\text{SRSA}}, \epsilon_{\text{SRSA}})$-SRSA assumption holds if for all $t_{\text{SRSA}}$-time adversaries $\mathcal{A}$

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

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

**Definition 6** (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 $i \in \{c, e, n, r, z\}$ be polynomials. We require that $|n|_2 = l_n$ and $|p'|_2 = |q'|_2 = l_n/2 - 1$. Furthermore, we assume that the $(t_{\text{SRSA}}, \epsilon_{\text{SRSA}})$-SRSA assumption holds. We let $u, v, w$ be public random generators of $QR_n$. When using a combining function $\mathsf{CMB}$, we assume that $k_{\mathsf{CMB}} \leftarrow \mathsf{CMB.KeyGen}(1^\kappa)$, $\mathcal{M}_{\mathsf{SIG}} = \mathcal{M}_{\mathsf{CMB}}$, $\mathcal{Z}_{\mathsf{CMB}} \subseteq [0; 2^{l_z} - 1]$, $\mathcal{R}_{\mathsf{CMB}} \subseteq [0; 2^{l_r} - 1]$, and that $k_{\mathsf{CMB}}$ is also part of $PK$. When using a chameleon hash function $\mathsf{CH}$, we assume that $PK_{\mathsf{CH}} \leftarrow \mathsf{CH.KeyGen}(1^\kappa)$, $\mathcal{M}_{\mathsf{SIG}} = \mathcal{M}_{\mathsf{CH}}$, $\mathcal{C}_{\mathsf{CH}} \subseteq [0; 2^{l_c} - 1]$, and that $PK_{\mathsf{CH}}$ is also part of $PK$. We let $E \subseteq [2^{l_e - 1}; 2^{l_e} - 1]$ denote the set of $l_e$-bit primes. Finally, we require that $l_c, l_z, l_r < l_e < l_n/2 - 1$.

### 2.8. *The Strong Diffie–Hellman Setting*

**Definition 7** (Bilinear Groups). Let $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ be groups of prime order $p$. The function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing if it holds that (1) for all $a \in \mathbb{G}_1$, $b \in \mathbb{G}_2$, and $x, y \in \mathbb{Z}_p$ we have $e(a^x, b^y) = e(a, b)^{xy}$ (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 $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ a bilinear group.

**Definition 8** (SDH Assumption (SDH)). Let $(\mathbb{G}_1, \hat{g}_1, \mathbb{G}_2, \hat{g}_2, \mathbb{G}_T, p, e)$ be a bilinear group. Let $l_p = l_p(\kappa)$ be a polynomial and $|p|_2 = l_p$. We say that the $(q_{\text{SDH}}, t_{\text{SDH}}, \epsilon_{\text{SDH}})$-SDH assumption holds if for all $t_{\text{SDH}}$-time attackers $\mathcal{A}$ that are given a $(q_{\text{SDH}} + 3)$-tuple of group elements $W = (g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^{q_{\text{SDH}}})}, g_2, g_2^x) \in \mathbb{G}_1^{q_{\text{SDH}}+1} \times \mathbb{G}_2^2$ it holds that

$$\Pr\big[(s, c) \leftarrow \mathcal{A}(W),\ c \in \mathbb{Z}_p,\ s \in \mathbb{G}_1,\ e\big(s, g_2^x g_2^c\big) = e(g_1, g_2)\big] \leq \epsilon_{\text{SDH}},$$

where the probability is over the random choices of the generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $x \in \mathbb{Z}_p$ and the random bits of $\mathcal{A}$.

**Definition 9** (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 $|p|_2 = l_p$. $\mathsf{SIG.KeyGen}(1^\kappa)$ chooses $x \xleftarrow{\$} \mathbb{Z}_p$ and outputs $(SK = x, PK = g_2^x)$. We assume that the $(q_{\text{SDH}}, t_{\text{SDH}}, \epsilon_{\text{SDH}})$-SDH assumption holds. Finally, we suppose that the values $a, b, c \in \mathbb{G}_1$ are public random generators of $\mathbb{G}_1$. In case we use a combining function $\mathsf{CMB}$, we assume that $\mathcal{M}_{\mathsf{SIG}} = \mathcal{M}_{\mathsf{CMB}}$, $\mathcal{Z}_{\mathsf{CMB}} \subseteq \mathbb{Z}_p$, $\mathcal{R}_{\mathsf{CMB}} \subseteq \mathbb{Z}_p$, and that $k_{\mathsf{CMB}}$ is also part of $PK$. When using a chameleon hash function $\mathsf{CH}$, we assume that $\mathcal{M}_{\mathsf{SIG}} = \mathcal{M}_{\mathsf{CH}}$, $PK_{\mathsf{CH}} \leftarrow \mathsf{CH.KeyGen}(1^\kappa)$, $\mathcal{C}_{\mathsf{CH}} \subseteq \mathbb{Z}_p$, and that $PK_{\mathsf{CH}}$ is also part of $PK$.

**Table 3.** 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<br>$e \xleftarrow{\$} E,\ r \xleftarrow{\$} \mathcal{R},\ \sigma = (r, s, e)$ | SDH setting<br>$t \xleftarrow{\$} \mathbb{Z}_p,\ r \xleftarrow{\$} \mathcal{R},\ \sigma = (r, s, t)$ |
|---|---|---|---|
| Chameleon hash | Sign | $s = (uv^{ch(r,m)})^{\frac{1}{e}}$ | $s = (ab^{ch(r,m)})^{\frac{1}{x+t}}$ |
| | Verify | $s^e \stackrel{?}{=} uv^{ch(r,m)},\ e$ odd?, $\lvert e\rvert = l_e$? | $e(s, PKg_2^t) \stackrel{?}{=} e(ab^{ch(r,m)}, g_2)$ |
| Combining | Sign | $s = (uv^r w^{z(r,m)})^{\frac{1}{e}}$ | $s = (ab^r c^{z(r,m)})^{\frac{1}{x+t}}$ |
| | Verify | $s^e \stackrel{?}{=} uv^r w^{z(r,m)},\ e$ odd?, $\lvert e\rvert = l_e$? | $e(s, PKg_2^t) \stackrel{?}{=} e(ab^r c^{z(r,m)}, g_2)$ |

## 3. 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. We provide an overview in Table 3.

### 3.1. *SRSA-Based Combining Signature Scheme* ($\mathcal{S}_{\mathrm{CMB,SRSA}}$)

In the SRSA setting, $\mathsf{SIG.Sign}(SK, m)$ randomly chooses $r \in \mathcal{R}_{\mathsf{CMB}}$ and $e \in E$ and computes a signature $\sigma = (r, s, e)$ on message $m$ with $s = (uv^r w^{z(r,m)})^{\frac{1}{e}}$. Let us now show, that our construction indeed generalizes the claimed signature schemes. Observe that we can easily obtain the Fischlin scheme [10] when we instantiate the combining function with $\mathsf{CMB}_2$ of Table 2. Furthermore, using $\mathsf{CMB}_3$ we can also get the Camenisch–Lysyanskaya scheme [4] with *smaller* parameter sizes. This becomes obvious if we substitute $v$ by $v' = vw$ as $uv^r w^{r+m} = u(vw)^r w^m = u(v')^r w^m$. Let us explain in more detail how our parameter choices deviate from [4]. 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 $l_m \leq l_r < l_e < l_n/2 - 1$. For comparable security, we could set $l_r = 320$, $l_s = 1024$, and $l_e = 321$ for a probability $\epsilon_{\mathsf{CMB}} = 2^{-160}$. 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 exponentiations—than the original scheme. We stress that this improvement does not even exploit our new security proof. A tight security proof accounts for considerable, additional efficiency. We note that when we use our variant of 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 (the revised) scheme of Zhu [14,23,24]. By Lemma 2, 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 $s^e \stackrel{?}{=} uv^r w^{z(r,m)}$, if $|e|_2 = l_e$, and if $e$ is odd.

### 3.2. *SDH-based Combining Signature Scheme* ($\mathcal{S}_{\mathrm{CMB,SDH}}$)

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 [5] and proven secure in [1,18]. Similar to $\mathcal{S}_{\mathrm{CMB,SRSA}}$, we obtain the SDH-based Camenisch–Lysyanskaya scheme when instantiating the combining function with $\mathsf{CMB}_1$. In the same way, we can also get SDH-based variants of the Fischlin signature scheme (using $\mathsf{CMB}_2$) and of Zhu's scheme (using Lemma 2). In the SDH-based combining scheme, $\mathsf{SIG.Sign}(SK, m)$ at first chooses a random $r \in \mathcal{R}_{\mathsf{CMB}}$ and a random $t \in \mathcal{Z}_{\mathsf{CMB}}$. It then computes the signature $\sigma = (r, s, t)$ with $s = (ab^r c^{z(r,m)})^{\frac{1}{x+t}}$. Given a signature $\sigma = (r, s, t)$, $\mathsf{SIG.Verify}(PK, m, \sigma)$ checks if $e(s, PKg_2^t) \stackrel{?}{=} e(ab^r c^{z(r,m)}, g_2)$.

### 3.3. *SRSA-Based Chameleon Hash Signature Scheme* ($\mathcal{S}_{\mathrm{CH,SRSA}}$)

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 $\mathsf{CH}$ using $\mathsf{CH.KeyGen}(1^\kappa)$. The value $PK_{\mathsf{CH}}$ is added to the scheme's public key $PK$. ($SK_{\mathsf{CH}}$ is not required. However, it may be useful when turning the signature scheme into an online-offline signature scheme [22].) The signature generation algorithm $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}_{\mathsf{CH}}$ and a random prime $e \in E$. It then outputs the signature $\sigma = (r, s, e)$ on a message $m$ where $s = (uv^{ch(r,m)})^{\frac{1}{e}}$. 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 $s^e \stackrel{?}{=} uv^{ch(r,m)}$.

### 3.4. *SDH-Based Chameleon Hash Signature Scheme* ($\mathcal{S}_{\mathrm{CH,SDH}}$)

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 $PK_{\mathsf{CH}}$ of a chameleon hash function to $PK$. In the SDH setting, $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}_{\mathsf{CH}}$ and a random $t \in \mathcal{Z}_{\mathsf{CH}}$. Using $SK = x$, it then outputs the signature $\sigma$ on $m$ as $\sigma = (r, s, t)$ where $s = (ab^{ch(r,m)})^{\frac{1}{x+t}}$. To verify a given signature $\sigma = (r, s, t)$ on $m$, $\mathsf{SIG.Verify}(PK, m, \sigma)$ checks if $e(s, PKg_2^t) \stackrel{?}{=} e(ab^{ch(r,m)}, g_2)$. A suitable chameleon hash function can for example be found in [15].

### 3.5. *The Cramer–Shoup Signature Scheme* ($\mathcal{S}_{\mathrm{CS,SRSA}}$)

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

- $\mathsf{SIG.KeyGen}(1^\kappa)$ additionally computes a random $l_e$-bit prime $\tilde{e}$. The secret key is $SK = (p, q)$ the public key is $PK = (n, \tilde{e})$.
- $\mathsf{SIG.Sign}(SK, m)$ first chooses a random $r \in QR_n$ and evaluates (the chameleon hash function, as shown below) $c = r^{\tilde{e}}/v^{h(m)} \bmod n$. Then it draws a random $l_e$-bit prime $e \neq \tilde{e}$ and computes the value $s = (uv^{h(c)})^{\frac{1}{e}} \bmod n$. The signature is $\sigma = (r, s, e)$.

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

It might be not obvious that the underlying structure of the Cramer–Shoup signature scheme is similar to that of the schemes in the chameleon hash class. To explain this we show that $c = r^{\tilde{e}}/v^{h(m)} \bmod n$ indeed constitutes a chameleon hash function. We emphasize that this chameleon hash function uses the same modulus $n$ (and the same generator $v$) that is also required for computing $s$. As a consequence its security cannot be based on the difficulty of factoring. However, security can still be based on the SRSA-assumption.

**Lemma 3.** *In the SRSA setting, let* $\mathsf{H} = (\mathsf{H.KeyGen}, \mathsf{H.Eval})$ *be a* $(t_{\mathsf{H}}, \epsilon_{\mathsf{H}})$*-collision-resistant hash function with message space* $\mathcal{M}_{\mathsf{H}} = \{0,1\}^*$ *and output space* $\mathcal{H}_{\mathsf{H}} = \{0,1\}^{l_c}$*. Let* $k_{\mathsf{H}}$ *be generated as* $k_{\mathsf{H}} \leftarrow \mathsf{H.KeyGen}(1^\kappa)$ *and* $\tilde{e} \in E$ *be a prime with* $l_c < l_e < l_n/2 - 1$*. Set* $PK_{\mathsf{CH}} = (k_{\mathsf{H}}, e, n, v = v'^{\tilde{e}} \bmod n)$ *and* $SK_{\mathsf{CH}} = v'$ *for some random* $v' \in QR_n$*. Then,* $ch(r,m) = h(v^{-h(m)}r^{\tilde{e}} \bmod n)$*,* $ch^{-1}(r,m,m') = r \cdot (v')^{(h(m')-h(m))} \bmod n$*, and the algorithms to set up* $SK_{\mathsf{CH}}$ *and* $PK_{\mathsf{CH}}$ *constitute a chameleon hash function with* $\mathcal{M}_{\mathsf{CH}} = \{0,1\}^*$*,* $\mathcal{R}_{\mathsf{CH}} = QR_n$*, and* $\mathcal{C}_{\mathsf{CH}} = \{0,1\}^{l_c}$ *that is* $(t, 3\epsilon_{\mathsf{H}}/2 + 3\epsilon_{\mathrm{SRSA}})$ *collision-resistant with* $t \approx \min\{t_{\mathsf{H}}, t_{\mathrm{SRSA}}\}$*.*

**Proof.** First, given $SK_{\mathsf{CH}}$, $r \in \mathcal{R}_{\mathsf{CH}}$, and $m, m' \in \mathcal{M}_{\mathsf{CH}}$, we can easily find $r' \in \mathcal{R}_{\mathsf{CH}}$ such that $ch(r,m) = ch(r',m')$ by computing $r' = r \cdot (v')^{(h(m')-h(m))} \bmod n$.

Second, observe that if $r$ is chosen uniformly at random from $QR_n$, $v^{-h(m)}r^{\tilde{e}} \bmod n$ is also distributed uniformly at random for all $m \in \mathcal{M}_{\mathsf{CH}}$. Therefore, $h(v^{-h(m)}r^{\tilde{e}} \bmod n)$ is equally distributed for every $m \in \mathcal{M}_{\mathsf{CH}}$.

Third, for contradiction assume a $t \approx \min\{t_{\mathsf{H}}, t_{\mathrm{SRSA}}\}$-time attacker $\mathcal{A}$ that can find $(m,r), (m',r')$ with $m \neq m'$ such that $h(v^{-h(m)}r^{\tilde{e}} \bmod n) = h(v^{-h(m')}(r')^{\tilde{e}} \bmod n)$ with probability better than $3/2\epsilon_{\mathsf{H}} + 3\epsilon_{\mathrm{SRSA}}$. Then, we can construct an algorithm $\mathcal{B}$ that breaks the SRSA assumption or the security of the hash function. $\mathcal{B}$ at first guesses with probability $\geq 1/3$ if $\mathcal{A}$ outputs a collision such that (1) $h(m) = h(m')$, (2) $v^{-h(m)}r^{\tilde{e}} \bmod n \neq v^{-h(m')}(r')^{\tilde{e}} \bmod n$ and $h(m) \neq h(m')$ or (3) $v^{-h(m)}r^{\tilde{e}} \bmod n = v^{-h(m')}(r')^{\tilde{e}} \bmod n$ and $h(m) \neq h(m')$. In the first two cases, $\mathcal{B}$ can output a collision for the hash function with probability better than $\epsilon_{\mathsf{H}}$. In the last case, $\mathcal{B}$ can break an RSAchallenge $(\hat{u} \bmod \hat{n}, \hat{n}, \hat{e})$ with probability better than $\epsilon_{\mathrm{SRSA}}$: $\mathcal{B}$ just inputs $(v := u, \tilde{e} = \hat{e}, n = \hat{n})$ into $\mathcal{A}$. Since for the output of $\mathcal{A}$ it holds that $v^{h(m')-h(m)} = (r/r')^{\tilde{e}} \bmod n$ and because $|h(m') - h(m)| < \tilde{e}$ and $\tilde{e}$ is prime, $\mathcal{B}$ can compute $a, b \in \mathbb{Z}$ such that $\gcd((h(m') - h(m)), \tilde{e}) = a(h(m') - h(m)) + b\tilde{e} = 1$. It can thus find a solution to the RSAchallenge as $(r/r')^a v^b \bmod n$. All cases contradict our starting assumptions which implies that $\mathcal{A}$ cannot exists, what finally proves Lemma 3. □

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 parts of the key material of its chameleon hash function are not chosen independently of the rest of the signature scheme. In particular, the chameleon hash function uses the same RSA modulus $n$ 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 Sect. 7.

## 4. Security

**Theorem 1.** *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 (with modified parameter sizes), 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. In Sect. 5, we present a *full* proof of security for $\mathcal{S}_{\mathrm{CMB,SRSA}}$, which seems to us to be the technically most involved SRSA-based reduction. The proof of $\mathcal{S}_{\mathrm{CMB,SDH}}$ proceeds analogously and appears in Sect. 6. We then show how to transfer our technique to $\mathcal{S}_{\mathrm{CH}}$ in Sect. 7 by providing a full proof of security of the Cramer–Shoup signature scheme. In the following, we only concentrate on the core idea of our new proofs. When formally proving the schemes secure, we additionally have to put some effort into guaranteeing that the public keys produced by the simulator are indistinguishable from a real public key. To this end the simulator has to randomize certain group elements via exponentiation by random values. Fortunately, this does not pose a serious threat to the overall proof strategy.

### 4.1. *The SRSA-Based Combining Signature Scheme*

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 $m_1, \ldots, m_q \in \mathcal{M}_{\mathsf{SIG}}$. As a response to each query $m_i$ with $i \in [1; q]$, $\mathcal{A}$ receives a corresponding signature $\sigma_i = (r_i, s_i, e_i) \in \mathcal{R}_{\mathsf{CMB}} \times QR_n \times E$. Recall that the existing security proofs for schemes of the combining class (e.g. [10]) 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 $\gcd(e^*, \prod_{i=1}^{q} e_i) \neq 1$.[2] 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 in [4,9,10,23,24] work as follows: the simulator $\mathcal{B}$ at first guesses $j \xleftarrow{\$} \{1, \ldots, q\}$. 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 $i \in \{1, \ldots, q\} \setminus \{j\}$), $\mathcal{B}$ 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 one choice of $r_i$ such that $\mathcal{B}$ can simulate

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

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.

We exploit that the $r_i$ and the primes $e_i$ are chosen independently at random by the signer. So, they can be specified *prior* to the signature queries in the simulation. For simplicity, assume that $\mathcal{B}$ can set up $\mathcal{A}$ 's input parameters such that the verification of a signature $\sigma = (r, s, e)$ always reduces to

$$s^e = \hat{u}^{f(r)} \bmod n. \tag{1}$$

Suppose that neither $\hat{u}$ nor $f : \mathcal{R}_{\mathsf{CMB}} \to \mathbb{N}$ are ever revealed to $\mathcal{A}$. 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

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

In the literature this computation is sometimes referred to as Shamir's trick. $\mathcal{B}$'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 $f$: the above requirements imply that we must have $e_i | f(r) \Leftrightarrow r = r_i$. Additionally $f$ needs to be linear such that it can be embedded in the exponents of two group elements ($u, v$ in the chameleon hash class). We now present our construction of $f$ and argue that it perfectly fulfills all requirements. We define $f : [0; 2^{l_e - 1} - 1] \to \mathbb{Z}$ as

$$f(r) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j - r \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j, \tag{2}$$

for $r_1, \ldots, r_q \in \mathcal{R}_{\mathsf{CMB}}$. Furthermore, $e_1, \ldots, e_q \in E$ must be *distinct* primes.

**Lemma 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 $r_1, \ldots, r_q \in [0; 2^{l_e - 1} - 1]$. Then we have for the above definition of $f$:*

$$e_k | f(r) \quad \Leftrightarrow \quad r = r_k$$

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

**Proof.** For convenience define $\bar{e}_i = \prod_{j=1, j\neq i}^{q} e_j$ for $i \in [1; q]$. Observe that because the $e_i$ are all distinct primes we get for $i, i' \in [1; q]$ that

$$\bar{e}_i = 0 \bmod e_{i'} \quad \Leftrightarrow \quad i' \neq i.$$

Using this notation, we have for all $k \in [1; q]$ that

$$f(r) = (r_k - r)\bar{e}_k \bmod e_k.$$

For $r = r_k$ we immediately get $f(r_k) = 0 \bmod e_k$. Now, assume $r \neq r_k$. Since the $e_i$ are distinct primes and as $|r_k - r| < e_k$, it holds that

$$f(r) = (r_k - r)\bar{e}_k \neq 0 \bmod e_k,$$

which proves the lemma. If we assume that $|r_i|_2 < l_e + 1$ for all $i \in [1; q]$ we can also have $r_1, \ldots, r_q \in [-2^{l_e}; 2^{l_e} - 1]$. □

### 4.2. *The SDH-Based Combining Signature Scheme*

Under the SDH assumption, the situation is very similar. Here we also analyze three possible types of forgeries $(m^*, (r^*, s^*, t^*))$: either (1) $t^* \notin \{t_1, \ldots, t_q\}$ or (2) $t^* = t_i$ with $i \in [1; q]$ but $r^* \neq r_i$ or (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 $(\hat{g}_1, \hat{g}_1^x, \hat{g}_1^{(x^2)}, \ldots, \hat{g}_1^{(x^q)}, g_2, g_2^x)$. This time, $\mathcal{B}$ chooses $PK = g_2^x$. In the SDH setting, (1) transfers to

$$e\big(s, PKg_2^t\big) = e\big(\hat{g}_1^{f(r,x)}, g_2\big) \quad \Leftrightarrow \quad s^{x+t} = \hat{g}_1^{f(r,x)}. \tag{3}$$

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 $f(r, x)$ with $f : \mathbb{Z}_p \times \mathbb{Z}_p[x] \to \mathbb{Z}_p[x]$ to map polynomials of maximal degree $q$ in indeterminate $x$ to polynomials of maximal degree $q$ as

$$f(r, x) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1 \\ j\neq i}}^{q} (x + t_j) - r \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j\neq i}}^{q} (x + t_j), \tag{4}$$

for $r_1, \ldots, r_q \in \mathcal{R}_{\mathsf{CMB}}$ and *distinct* $t_1, \ldots, t_q \in \mathbb{Z}_p$. (Recall that $\mathcal{R}_{\mathsf{CMB}} \subseteq \mathbb{Z}_p$ so $f$ is useful if we choose $r_i$ from $\mathcal{R}_{\mathsf{CMB}}$.) Using the SDH challenge, $\mathcal{B}$ can easily compute $\hat{g}_1^{f(r,x)}$ since $f(r, x)$ has maximal degree $q$. In the following, we will, given a degree $q$ polynomial $f(r, x)$ and $(x + t)$, often apply long division to compute a polynomial $f'(r, x)$ of degree $q - 1$ and $D \in \mathbb{Z}_p$ such that $f(r, x) = f'(r, x)(x + t) + D$. We use $(x + t) | f(r, x)$ to denote that long division results in $D = 0$. Similarly, we use $(x + t) \not| f(r, x)$ to denote that $D \neq 0$.

**Lemma 5.** *Let a set $T = \{t_1, \ldots, t_q\} \subseteq \mathbb{Z}_p$ and $q$ values $r_1, \ldots, r_q \in \mathbb{Z}_p$ be given. Then the above definition of $f(r, x)$ (4) guarantees that*

$$(x + t_k) | f(r, x) \quad \Leftrightarrow \quad r = r_k$$

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

**Proof.** Define $\bar{t}_i := \prod_{k=1,k\neq i}^{q}(t_k + x)$ for $i \in [1;q]$. Since the $t_i$ are all distinct, we get for $i, i' \in [1;q]$ that

$$(x + t_{i'}) | \bar{t}_i \quad \Leftrightarrow \quad i' \neq i.$$

Also, observe that for every $k \in [1;q]$ we always have

$$(x + t_k) | \big(f(r,x) - (r_k - r)\bar{t}_k\big).$$

From this we immediately get for $r = r_k$ that $(x + t_k) | f(r_k, x)$. On the other hand, for $r \neq r_k$, we get that $(x + t_k) \nmid (r_k - r)\bar{t}_k$. However, this also shows that we cannot have $(x + t_k) | f(r,x)$ as otherwise $(x + t_k) \nmid (f(r,x) - (r_k - r)\bar{t}_k)$. So it must hold that $(x + t_k) \nmid f(r)$ which concludes the proof. □

It remains to show how to extract a solution to the SDH challenge from $\mathcal{A}$'s forgery. If $r \neq r_k$, then long division gives us $D \in \mathbb{Z}$ with $D \neq 0$ and a new polynomial $\tilde{f}_{t_k}(r,x)$ with coefficients in $\mathbb{Z}$ such that $f(r,x) = \tilde{f}_{t_k}(r,x)(x + t_k) + D$. Also, $\tilde{f}_{t_k}(r,x)$ has degree $<q$ and can be evaluated with the help of the SDH challenge. Similar to the SRSA class, we can find a solution to the SDH challenge from $\mathcal{A}$'s forgery as

$$\Big(\big((s^*)\hat{g}_1^{-\tilde{f}_{t^*}(r^*,x)}\big)^{1/D} = \hat{g}_1^{1/(x+t^*)},\ t^*\Big).$$

### 4.3. *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 again straight-forward and very similar to the corresponding proof 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 $f(c)$ ($f(c,x)$). Recall Sect. 4 where we analyzed the equations $s^e = \hat{u}^{f(r)}$ and $f(r) = \sum_{i=1}^{q} r_i \prod_{j=1,j\neq i}^{q} e_j - r\sum_{i=1}^{q}\prod_{j=1,j\neq i}^{q} e_j$ in the SRSA setting (and $s^{x+t} = \hat{g}_1^{f(r,x)}$ and $f(r,x) = \sum_{i=1}^{q} r_i \prod_{j=1,j\neq i}^{q}(x + t_j) - r\sum_{i=1}^{q}\prod_{j=1,j\neq i}^{q}(x + t_j)$ in the SDH setting). In the proof of the combining class the $r_i$ are random values that can be specified prior to the simulation phase. In the proof of the chameleon hash class we take a similar approach. However, now we use the $c_i$, the output values of the chameleon hash function to set up $f$. In the initialization phase of the proof we choose $q$ random input pairs $(m'_i, r'_i) \in \mathcal{M}_{\mathsf{CH}} \times \mathcal{R}_{\mathsf{CH}}$, $i \in [1;q]$ 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 $l_e$-bit primes (random values $t_1, \ldots, t_q \in \mathbb{Z}_p$) as in the proofs of the combining class and exploit Lemma 4 (Lemma 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 $SK_{\mathsf{CH}}$ 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).

## 5. Security Analysis of $\mathcal{S}_{\mathsf{CMB,SRSA}}$

In the following, we provide a detailed security proof for the SRSA-based combining class. But first we establish a useful lemma on the number of primes with a given bit length.

**Lemma 6.** *Let $E$ be the set of primes in the interval $[2^{l_e-1}; 2^{l_e}-1]$. For $l_e \geq 33$*

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

**Proof.** Rosser presented explicit bounds for the prime counting function $\pi(x)$ in [20]. For $x \geq 55$, he showed that

$$\frac{x}{\ln(x)+2} \leq \pi(x) \leq \frac{x}{\ln(x)-4}.$$

For $l_e \geq 33 > 4 + \frac{20}{\ln(2)}$ we have in particular that (a) $\frac{10+2\ln(2)}{l_e \ln(2)} < 1/2$ and $2^{l_e} > 55$. At the same time, since $l_e > 9 > 1 + \frac{5}{\ln(2)}$, we get (b) $l_e \ln(2) > l_e \ln(2) - 4 - \ln(2) > 1$. Also, since $l_e > 7 > \frac{2}{1-\ln(2)}$ we have that (c) $l_e > l_e \ln(2) + 2$. We exploit that $\pi(2^{l_e}-1) = \pi(2^{l_e})$ and thus the lower bound for $\pi(2^{l_e})$ must also hold for $\pi(2^{l_e}-1)$. We can now bound the number of $l_e$-bit primes as

$$\begin{aligned}
|E| &\geq \frac{2^{l_e}}{l_e \ln(2)+2} - \frac{2^{l_e-1}}{(l_e-1)\ln(2)-4} = 2^{l_e-1} \cdot \frac{l_e \ln(2) - 10 - 2\ln(2)}{(l_e \ln(2)+2)(l_e \ln(2) - \ln(2) - 4)} \\
&\overset{(b)}{>} 2^{l_e-1} \cdot \frac{l_e \ln(2) - 10 - 2\ln(2)}{(l_e \ln(2)+2) l_e \ln(2)} = 2^{l_e-1} \cdot \frac{1 - \frac{10+2\ln(2)}{l_e \ln(2)}}{(l_e \ln(2)+2)} \\
&\overset{(a)}{>} \frac{2^{l_e-2}}{(l_e \ln(2)+2)} \overset{(c)}{>} \frac{2^{l_e-2}}{l_e}.
\end{aligned}$$

□

Now let us proceed to the first tight security proof. The proof is similar to the original proof of the Cramer–Shoup signature scheme.

**Lemma 7.** *Assume we work in the SRSA setting such that the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds and* CMB *is a $(t_{\mathsf{CMB}}, \epsilon_{\mathsf{CMB}}, \delta_{\mathsf{CMB}})$-combining function. Then, the combining signature class as presented in Sect. 3.1 is $(q, t, \epsilon)$-secure against adaptive chosen message attacks provided that*

$$q = q_{\mathrm{SRSA}}, \qquad \epsilon < \frac{9}{2}\epsilon_{\mathrm{SRSA}} + 3\epsilon_{\mathsf{CMB}} + 3q\delta_{\mathsf{CMB}} + \frac{3q^2 l_e}{2^{l_e-2}} + \frac{9}{2^{l_n/2-2}}, \qquad t \approx t_{\mathrm{SRSA}}.$$

The proof of Lemma 7 is the first step in the proof of Theorem 1. It implies that the original Fischlin and the Zhu's signature scheme as well as the modified Camenisch–Lysyanskaya are tightly secure against existential forgeries under adaptive chosen message attacks.

**Proof.** Assume that $\mathcal{A}$ is a forger that $(q, t, \epsilon)$-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 $t_{\mathrm{SRSA}}$ with advantage $\epsilon_{\mathrm{SRSA}}$. 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 7 then follows by a standard averaging 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.

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

$$\Pr[S_0] = \epsilon. \tag{5}$$

$\mathrm{Game}_1$ 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 $e_1, \ldots, e_q \xleftarrow{\$} E$ and three random elements $t_0', t_0'' \xleftarrow{\$} \mathbb{Z}_{(n-1)/4}$ and $t_0 \xleftarrow{\$} \mathbb{Z}_{3(n-1)/4}$. 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 $u = \hat{u}^{2t_0\bar{e}}$, $v = \hat{u}^{2t_0'\bar{e}}$, $w = \hat{u}^{2t_0''\bar{e}}$ 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

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

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

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)}. \tag{6}$$

$\mathrm{Game}_2$ Now, $\mathcal{B}$ simulates $\mathcal{O}(SK, \cdot)$ by answering $\mathcal{A}$'s signature queries. Subsequently, set $z_j = z(r_j, m_j)$ and $z^* = z(r^*, m^*)$. The simulator $\mathcal{B}$ sets $PK = (u, v, w, n)$ and for all $j \in \{1, \ldots, q\}$ it chooses a random $r_j \in \mathcal{R}_{\mathrm{CMB}}$ and outputs $\sigma_j = (r_j, s_j, e_j)$ with $s_j = (uv^{r_j}w^{z_j})^{\frac{1}{e_j}} = \hat{u}^{2(t_0 + t_0'r_j + t_0''z_j)\bar{e}_j}$. The distribution of the so computed values is

equal to the previous game and

$$\Pr[S_2] = \Pr[S_1]. \tag{7}$$

Game$_3$ Now, consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. Define $\hat{e} = (t_0 + t_0' r^* + t_0'' z^*)$. 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 $T + \tilde{t}_{0,2} p'q' = 0 \bmod e^*$. 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 SRSA challenge (using the techniques described below).
Assume there exists at least one such $\tilde{t}_{0,2}$. Then, we have that $T + \tilde{t}_{0,2} p'q' = 0 \bmod e^*$. Let us analyze the remaining possibilities $\tilde{t}_{0,2} \pm 1 \bmod 3$ and $\tilde{t}_{0,2} \pm 2 \bmod 3$ as $A = T + \tilde{t}_{0,2} p'q' \pm p'q' \bmod e^*$ and $B = T + \tilde{t}_{0,2} p'q' \pm 2p'q' \bmod e^*$. Since $\gcd(e^*, p'q') < e^*$, we know that $p'q' \neq 0 \bmod e^*$. As $T + \tilde{t}_{0,2} p'q' = 0 \bmod e^*$, 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

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

Finally, we have that

$$\Pr[S_3] \geq 2 \cdot \Pr[S_2]/3 \tag{8}$$

and

$$\Pr[S_3] = \epsilon_{\mathrm{SRSA}}. \tag{9}$$

Plugging in (5)–(9), we get that $\epsilon \leq \frac{3}{2}\epsilon_{\mathrm{SRSA}} + 3 \cdot 2^{2-l_n/2}$.

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

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)}. \tag{10}$$

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 pre-computed $r_j$ and $e_j$ and computing $s_j$ as

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

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

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|}. \tag{11}$$

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

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

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.

$$\Pr[S_3] = \epsilon_{\text{SRSA}}. \tag{12}$$

Summing up (10)–(12), 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 $z_1,\ldots,z_q \in \mathcal{Z}_{\mathsf{CMB}}$. 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 $u = \hat{u}^{2(t_0\bar{e}+\sum_{i=1}^{q} z_i\bar{e}_i)}$, $v = \hat{u}^{2t_0'\bar{e}}$, and $w = \hat{u}^{2(t_0''\bar{e}-\sum_{i=1}^{q}\bar{e}_i)}$.

$$\Pr[S_1] \geq \Pr[S_0] - 3\cdot 2^{-(l_n/2-2)}. \tag{13}$$

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

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|}. \tag{14}$$

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

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

To bound the probability of $\mathcal{B}$ to abort observe that the properties of the combining function guarantee that the $r_j$ are indistinguishable from uniformly random values over $\mathcal{R}_{\mathsf{CMB}}$ such that,

$$\Pr[S_3] \geq \Pr[S_2] - q\delta_{\mathsf{CMB}}. \tag{15}$$

When using $\mathsf{CMB}_1$, $\mathsf{CMB}_2$, or $\mathsf{CMB}_3$ the $r_j$ are even statistically close to uniform.

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 $i$-th message/signature pair. For all $t_{\mathsf{CMB}}$-time attackers such a collision happens with probability at most $\epsilon_{\mathsf{CMB}}$. Therefore,

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\mathsf{CMB}}. \tag{16}$$

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

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

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,

$$\Pr[S_4] = \epsilon_{\mathsf{SRSA}}. \tag{17}$$

Summing up (13)–(17), we get that $\epsilon \leq \epsilon_{\mathsf{SRSA}} + \epsilon_{\mathsf{CMB}} + q\delta_{\mathsf{CMB}} + q^2/|E| + 3 \cdot 2^{2-l_n/2}$. Applying Lemma 6 to bound $|E|$ completes the proof. □

*Remark 1.* As shown in Sect. 3.1 one can, using our terminology, view the Zhu signature scheme as a combining scheme that relies on $\mathsf{CMB}_3$ and Lemma 2. Note that the application of a hash function does not influence $\delta_{\mathsf{CMB}}$, solely $\epsilon_{\mathsf{CMB}}$ is affected. Therefore, for the combining function obtained as a sequential combination of a hash function and $\mathsf{CMB}_3$ to the input message, $\delta_{\mathsf{CMB}}$ is entirely determined by $\mathsf{CMB}_3$. So, let us in the following solely concentrate on $\mathsf{CMB}_3$. In the original, flawed proof in [23,24], Zhu uses parameter lengths with $l_r = l_m$. As a consequence we only get that $\delta_{\mathsf{CMB}} \leq 1$. This is problematic in the security proof as for the probability bound of the signature scheme we only know that $\epsilon \leq \epsilon_{\mathsf{SRSA}} + 3\epsilon_{\mathsf{CMB}} + 3q\delta_{\mathsf{CMB}} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}$. We thus cannot argue that $\epsilon$ is negligible and the signature scheme secure. Joye in [14] fixed this problem simply by using parameter sizes $l_r$ and $l_m$ with $l_r > l_m$ such that $\delta_{\mathsf{CMB}} \leq 2^{l_m - l_r}$ is negligible.

## 6. Security Analysis of $\mathcal{S}_{\mathsf{CMB,SDH}}$

In the following we show how to transfer the security proof of the combining class to the SDH setting.

**Lemma 8.** *Assume we work in the SDH setting such that the $(t_{SDH}, \epsilon_{SDH})$-SDH assumption holds and* CMB *is a $(t_{\mathsf{CMB}}, \epsilon_{\mathsf{CMB}}, \delta_{\mathsf{CMB}})$-combining function. Then, the combining signature class as presented in Sect. 3.1 is $(q, t, \epsilon)$-secure against adaptive chosen message attacks provided that*

$$q = q_{\mathrm{SDH}}, \qquad \epsilon \leq 3\epsilon_{\mathrm{SDH}} + 3\epsilon_{\mathsf{CMB}} + 3q\delta_{\mathsf{CMB}} + \frac{3q^2}{p}, \qquad t \approx t_{\mathrm{SDH}}.$$

*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 8 is the second step in the proof of Theorem 1. 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 $(m^*, (r^*, s^*, t^*))$. The simulator $\mathcal{B}$ is given $(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x)$.

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

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

$$\Pr[S_0] = \epsilon. \tag{18}$$

$\mathrm{Game}_1$ 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 $t_1, \ldots, t_q \xleftarrow{\$} \mathbb{Z}_p$ and three random elements $t_0, t_0', t_0'' \xleftarrow{\$} \mathbb{Z}_p$. 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 $a = g_1^{t_0\bar{t}}$, $b = g_1^{t_0'\bar{t}}$, $d = g_1^{t_0''\bar{t}}$
Since the $t_0, t_0', t_0''$ are chosen uniformly at random from $\mathbb{Z}_p$, we have

$$\Pr[S_1] = \Pr[S_0]. \tag{19}$$

$\mathrm{Game}_2$ 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 $r_j \in \mathcal{R}_{\mathsf{CMB}}$ and outputs $\sigma_j = (r_j, s_j, t_j)$ with $s_j = (ab^{r_j}d^{z_j})^{1/(x+t_j)} = g_1^{(t_0 + t_0'r_j + t_0''z_j)\bar{t}_j}$.

$$\Pr[S_2] = \Pr[S_1]. \tag{20}$$

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

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

which is equivalent to

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

$\mathcal{B}$ 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

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

Finally, we have that

$$\Pr[S_3] = \Pr[S_2] = \epsilon_{\mathrm{SDH}}. \tag{21}$$

Plugging in (18)–(21), we get that $\epsilon = \epsilon_{\mathrm{SDH}}$.

*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.

Game$_1$ First, $\mathcal{B}$ randomly chooses $q$ *distinct* elements $t_1, \ldots, t_q \in \mathbb{Z}_p$ and $q$ random elements $r_1, \ldots, r_q \in \mathcal{R}_{\mathsf{CMB}}$. Additionally, it chooses three random elements $t_0, t_0', t_0'' \in \mathbb{Z}_p$. Next, $\mathcal{B}$ computes $a = g_1^{(t_0 \bar{t} + \sum_{i=1}^q r_i \bar{t}_i)}$, $b = g_1^{(t_0' \bar{t} - \sum_{i=1}^q \bar{t}_i)}$, and $d = g_1^{t_0'' \bar{t}}$ Again,

$$\Pr[S_1] = \Pr[S_0]. \tag{22}$$

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 $\sigma_j = (r_j, s_j, t_j)$ using the pre-computed $r_j$ and $t_j$ and computing $s_j$ as

$$s_j = \left(ab^{r_j} d^{z_j}\right)^{1/(x+t_j)} = g_1^{(t_0+t_0' r_j + t_0'' z_j)\bar{t}_j + \sum_{i=1, i\neq j}^q (r_i - r_j)\bar{t}_{i,j}}, \tag{23}$$

$$\Pr[S_2] \geq \Pr[S_1] - q^2/p.$$

Game$_3$ Now consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, t^*))$. By assumption there is a $k \in \{1, \ldots, q\}$ with $t^* = t_k$ and $r_k \neq r^*$. Then we have that

$$s^* = \left(ab^{r^*} d^{z^*}\right)^{1/(x+t^*)} = \left(g_1^{(t_0+t_0' r^*+t_0'' z^*)\bar{t} + \sum_{i=1}^q (r_i - r^*)\bar{t}_i}\right)^{1/(x+t^*)}$$
$$= \left(g_1^{(t_0+t_0' r^*+t_0'' z^*)\bar{t} + \sum_{i=1, i\neq k}^q (r_i - r^*)\bar{t}_i + (r_k - r^*)\bar{t}_k}\right)^{1/(x+t_k)}. \tag{24}$$

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

$$\left(s^* g_1^{-((t_0+t_0' r^*+t_0'' z^*)\bar{t}_k + \sum_{i=1, i\neq k}^q (r_i - r^*)\bar{t}_{i,k} + (r_k - r^*) f'(x))}\right)^{1/D(r_k - r^*)} = g_1^{1/(x+t_k)} \tag{25}$$

constitutes a solution to the SDH challenge,

$$\Pr[S_3] = \Pr[S_2] = \epsilon_{\mathsf{SDH}}. \tag{26}$$

Summing up (22)–(26), we get that $\epsilon \leq \epsilon_{\mathsf{SDH}} + q^2/p$.

*Type III Forger* ($t^* = t_i$ 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 $q$ values $t_1, \ldots, t_q \in \mathbb{Z}_p$ and $q$ random values $z_1, \ldots, z_q \in \mathcal{Z}_{\mathsf{CMB}}$. 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 $c = g_1^{t_0''\bar{t}-\sum_{i=1}^{q}\bar{t}_i}$.

$$\Pr[S_1] = \Pr[S_0]. \tag{27}$$

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

$$\Pr[S_2] \geq \Pr[S_1] - q^2/p. \tag{28}$$

Game$_3$ 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 $r_j = \perp$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs the signature $\sigma_j = (r_j, s_j, t_j)$ with $s_j$ being computed as

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

We have,

$$\Pr[S_3] \geq \Pr[S_2] - q\delta_{\mathsf{CMB}}. \tag{29}$$

Game$_4$ 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 $t_{\mathsf{CMB}}$-time attackers this happens with probability at most $\epsilon_{\mathsf{CMB}}$. Therefore,

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\mathsf{CMB}}. \tag{30}$$

It now holds that

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

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

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

is a solution to the SDH challenge. Finally,

$$\Pr[S_4] = \epsilon_{\mathsf{SDH}}. \tag{31}$$

Summing up (27)–(31), we get that $\epsilon \leq \epsilon_{\mathsf{SDH}} + \epsilon_{\mathsf{CMB}} + q\delta_{\mathsf{CMB}} + q^2/p$. □

## 7. Security Analysis of the Cramer–Shoup Signature Scheme

The original proof in [9] 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 Sect. 4.1.

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 (1) and (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 pre-specified values. Since $f(r)$ is linear, $\mathcal{B}$ can efficiently embed it 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. In the following we provide a formal proof.

**Lemma 9.** *Assume the* $(t_{\mathrm{SRSA}}, \epsilon_{\mathrm{SRSA}})$*-SRSA assumption holds. Moreover, let* $\mathsf{H}$ *be a* $(t_{\mathsf{H}} = t_{\mathrm{SRSA}}, \epsilon_{\mathsf{H}})$*-collision-resistant hash function with message space* $\mathcal{M}_{\mathsf{H}} = \{0,1\}^*$ *and output space* $\mathcal{H}_{\mathsf{H}} = \{0,1\}^{l_c}$*. Then, the Cramer–Shoup signature scheme is* $(q, t, \epsilon)$*-secure against* Type III Forgers *provided that*

$$q = q_{\mathrm{SRSA}}, \qquad \epsilon < \epsilon_{\mathrm{SRSA}} + \epsilon_{\mathsf{H}} + \frac{q^2 l_e}{2^{l_e - 2}} + \frac{1}{2^{l_n/2 - 3}}, \qquad t \approx t_{\mathrm{SRSA}}.$$

The proof of Lemma 9 can easily be transferred to the chameleon hash class in general.

**Proof.**

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

$$\Pr[S_0] = \epsilon. \tag{32}$$

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

$$\Pr[S_1] \geq \Pr[S_0] - 2 \cdot 2^{-(l_n/2-2)}. \tag{33}$$

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

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|}. \tag{34}$$

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

$$\Pr[S_3] = \Pr[S_2]. \tag{35}$$

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

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\mathsf{H}}. \tag{36}$$

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

$$\begin{aligned} \left(s^*\right)^{e^*} &= u v^{h(c^*)} \bmod n \\ &= \hat{u}^{2\tilde{e}((t_0 - t_0' h(m^*))\bar{e} + \sum_{i=1}^{q} (h(c_i) - h(c^*))\bar{e}_i} \bmod n \end{aligned}$$

which is equivalent to

$$\left(\left(s^*\right)\hat{u}^{-2\tilde{e}((t_0 - t_0' h(m^*))\bar{e}_j + \sum_{i=1, i\neq j}^{q} (h(c_i) - h(c^*))\bar{e}_{i,j}}\right)^{e_j} = \hat{u}^{2\tilde{e}(h(c_j) - h(c^*))\bar{e}_j} \bmod n.$$

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

$$\hat{u}^{1/e_j} = \bar{u}^a \left(\left(s^*\right)\hat{u}^{-2\tilde{e}((t_0 - t_0' h(m^*))\bar{e}_j + \sum_{i=1, i\neq j}^{q} (h(c_i) - h(c^*))\bar{e}_{i,j}}\right)^b.$$

We finally have

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

Applying Lemma 6 to lower bound $|E|$ completes the proof. □

## 8. Conclusion

In this paper, we have presented the first tight security proofs for a large class of signature schemes which are secure under the SRSA and the SDH assumption in the standard model. Our results can easily be extended to signature schemes for message blocks (as defined in [4,5]), where we can even use distinct combining functions for each message block. An interesting open question is whether there exist a tight security proof for the SRSA-based signature schemes without random oracles by Hofheinz and Kiltz [13]. This scheme outputs very small signatures if we ignore the influence of the looseness of the security reduction on the modulus size $n$—a simplification that is common in the literature. However, if we account for the tightness loss, the modulus size $n$ should considerably be increased to achieve the same security level and it seems that existing tightly secure signature schemes outperform the Hofheinz-Kiltz scheme in terms of signature size then.

## References

[1] M.H. Au, W. Susilo, Y. Mu, Constant-size dynamic $k$-TAA, in *SCN*, ed. by R. De Prisco, M. Yung. Lecture Notes in Computer Science, vol. 4116 (Springer, Berlin, 2006), pp. 111–125

[2] D.J. Bernstein, Proving tight security for Rabin–Williams signatures, in *EUROCRYPT*, ed. by N.P. Smart. Lecture Notes in Computer Science, vol. 4965 (Springer, Berlin, 2008), pp. 70–87

[3] D. Boneh, X. Boyen, Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptol.* **21**(2), 149–177 (2008)

[4] J. Camenisch, A. Lysyanskaya, A signature scheme with efficient protocols, in *SCN*, ed. by S. Cimato, C. Galdi, G. Persiano. Lecture Notes in Computer Science, vol. 2576 (Springer, Berlin, 2002), pp. 268–289

[5] J. Camenisch, A. Lysyanskaya, Signature schemes and anonymous credentials from bilinear maps, in *CRYPTO*, ed. by M.K. Franklin. Lecture Notes in Computer Science, vol. 3152 (Springer, Berlin, 2004), pp. 56–72

[6] D. Catalano, D. Fiore, B. Warinschi, Adaptive pseudo-free groups and applications, in Paterson [19], pp. 207–223

[7] B. Chevallier-Mames, M. Joye, A practical and tightly secure signature scheme without hash function, in *CT-RSA*, ed. by M. Abe. Lecture Notes in Computer Science, vol. 4377 (Springer, Berlin, 2007), pp. 339–356

[8] J.-S. Coron, D. Naccache, Security analysis of the Gennaro–Halevi–Rabin signature scheme, in *EUROCRYPT*, ed. by B. Preneel. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 91–101

[9] R. Cramer, V. Shoup, Signature schemes based on the Strong RSA assumption. *ACM Trans. Inf. Syst. Secur.* **3**(3), 161–185 (2000)

[10] M. Fischlin, The Cramer–Shoup Strong-RSA signature scheme revisited, in *Public Key Cryptography*, ed. by Y. Desmedt. Lecture Notes in Computer Science, vol. 2567 (Springer, Berlin, 2003), pp. 116–129

[11] R. Gennaro, S. Halevi, T. Rabin, Secure hash-and-sign signatures without the random oracle, in *EUROCRYPT* (1999), pp. 123–139

[12] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)

[13] D. Hofheinz, E. Kiltz, Programmable hash functions and their applications, in *CRYPTO*, ed. by D. Wagner. Lecture Notes in Computer Science, vol. 5157 (Springer, Berlin, 2008), pp. 21–38
[14] M. Joye, How (Not) to design Strong-RSA signatures. *Des. Codes Cryptogr.* **59**(1–3), 169–182 (2011)
[15] H. Krawczyk, T. Rabin, Chameleon signatures, in *NDSS* (The Internet Society, 2000)
[16] A. Lysyanskaya, R.L. Rivest, A. Sahai, S. Wolf, Pseudonym systems, in *Selected Areas in Cryptography*, ed. by H.M. Heys, C.M. Adams. Lecture Notes in Computer Science, vol. 1758 (Springer, Berlin, 1999), pp. 184–199
[17] D. Naccache, D. Pointcheval, J. Stern, Twin signatures: an alternative to the hash-and-sign paradigm, in *ACM Conference on Computer and Communications Security* (2001), pp. 20–27
[18] T. Okamoto, Efficient blind and partially blind signatures without random oracles, in *TCC*, ed. by S. Halevi, T. Rabin. Lecture Notes in Computer Science, vol. 3876 (Springer, Berlin, 2006), pp. 80–99
[19] K.G. Paterson (ed.), *Proceedings Advances in Cryptology—EUROCRYPT 2011—30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Tallinn, Estonia, May 15–19, 2011. Lecture Notes in Computer Science, vol. 6632 (Springer, Berlin, 2011)
[20] B. Rosser, Explicit bounds for some functions of prime numbers. *Am. J. Math.* **63**(1), 211–232 (1941)
[21] S. Schäge, Tight proofs for signature schemes without random oracles, in Paterson [19], pp. 189–206
[22] A. Shamir, Y. Tauman, Improved online/offline signature schemes, in *CRYPTO*, ed. by J. Kilian. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 355–367
[23] H. Zhu, New digital signature scheme attaining immunity to adaptive-chosen message attack. *Chin. J. Electron.* **10**(4), 484–486 (2001)
[24] H. Zhu, A formal proof of Zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155 (2003). http://eprint.iacr.org/