# Short Signatures from Weaker Assumptions

Dennis Hofheinz[1], Tibor Jager[2], and Eike Kiltz[2]

[1] Institut für Kryptographie und Sicherheit,
Karlsruhe Institute of Technology, Germany
Dennis.Hofheinz@kit.edu
[2] Horst-Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{tibor.jager,eike.kiltz}@rub.de

**Abstract.** We provide constructions of $(m, 1)$-programmable hash functions (PHFs) for $m \geq 2$. Mimicking certain programmability properties of random oracles, PHFs can, e.g., be plugged into the generic constructions by Hofheinz and Kiltz (J. Cryptol. 2011) to yield digital signature schemes from the strong RSA and strong $q$-Diffie-Hellman assumptions. As another application of PHFs, we propose new and efficient constructions of digital signature schemes from weaker assumptions, i.e., from the (standard, non-strong) RSA and the (standard, non-strong) $q$-Diffie-Hellman assumptions.

The resulting signature schemes offer interesting tradeoffs between efficiency/signature length and the size of the public-keys. For example, our $q$-Diffie-Hellman signatures can be as short as 200 bits; the signing algorithm of our Strong RSA signature scheme can be as efficient as the one in RSA full domain hash; compared to previous constructions, our RSA signatures are shorter (by a factor of roughly 2) and we obtain a considerable efficiency improvement (by an even larger factor). All our constructions are in the standard model, i.e., without random oracles.

**Keywords:** digital signatures, RSA assumption, $q$-DH assumption, programmable hash functions.

## 1 Introduction

Digital Signatures are one of the most fundamental cryptographic primitives. They are used as a building block in numerous high-level cryptographic protocols. Practical signature schemes are known whose security is based on relatively mild intractability assumptions such as the RSA [6] or the (bilinear) Computational Diffie-Hellman (CDH) assumption [13]. However, their security can only be proved in the random oracle model [5] with all its limitations (e.g., [17,26]).

Standard Model Signatures. Signature schemes in the standard model (i.e., without using random oracles) are often considerably less efficient or based on much stronger assumptions. While tree-based signature schemes can be built from any one-way function [48], these constructions are far from practical. On the other hand, "Hash-and-sign" signatures are considerably more efficient, but the

D.H. Lee and X. Wang (Eds.): ASIACRYPT 2011, LNCS 7073, pp. 647–666, 2011.

most efficient of these schemes rely on specific "strong" number theoretic hardness assumptions which we call Strong $q$-assumptions.[1] In Strong $q$-assumptions, an adversary is provided with a polynomial number of random "solved instances" and has to compute a new solved instance *of its choice*. For example, the schemes in [23,29,28,36,38,50] are based on the Strong (or, Flexible) RSA assumption and the schemes in [11,38,50] are based on the Strong $q$-Diffie-Hellman assumption. Both assumptions are considerably stronger than their "non-strong" counterparts (i.e., the $q$-Diffie-Hellman and the RSA assumptions, respectively), in which an adversary has to solve a *given, fixed instance*. (See the full version of this paper [34] for a discussion of the exact difference between strong and non-strong assumptions.)

Programmable Hash Functions. In order to mimic certain "programmability properties" of random oracles, Hofheinz and Kiltz [38] introduced the combinatorial concept of programmable hash functions (PHF). (See Section 3 for a formal definition.) Among a number of other applications, they used PHFs as a building block for efficient and short hash-and-sign signatures based on the Strong RSA and the Strong $q$-Diffie-Hellman assumptions. Concretely, signatures in the Strong RSA based HK signature scheme $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}]$ are of the form $\mathsf{sig}(M) = (\mathsf{H}(M)^{1/e} \bmod N, e)$, where $N = pq$ is a public RSA modulus, $\mathsf{H}(\cdot)$ is a $(m, 1)$-PHF, and $e$ is a *short* prime (chosen at random during the signing process). A given HK signature $(\sigma, e)$ is verified by checking if $\sigma^e = \mathsf{H}(M) \bmod N$. The efficiency of the HK signature scheme is dominated by the time needed to generate the prime $e$, which (as shown in [38]) depends on the parameter $m$ of the PHF: the bigger $m$, the smaller $e$ and consequently the more efficient is the signing process.[2] Over bilinear groups there exists a similar construction, $\mathsf{Sig}_{\mathsf{S}\text{-}q\text{-}\mathsf{DH}}[\mathsf{H}]$, whose security is based on the Strong $q$-DH assumption. The main disadvantages of HK signatures is that their security relies on Strong assumptions, i.e., on the Strong RSA (Strong $q$-DH) and not on the standard RSA ($q$-DH) assumption.

RSA signatures. As a step towards practical signatures from the (standard) RSA assumption, Hohenberger and Waters [40,39] proposed the first hash-and-sign signature scheme (HW signatures) whose security is based on the RSA assumption. HW signatures are computed as $\mathsf{sig}(M) = g^{1/\mathsf{P}(M)} \bmod N$, where $g \in \mathbb{Z}_N^*$ is a public element and $\mathsf{P}(M) = e_1 \cdot \ldots \cdot e_{|M|}$ is the product of $|M|$ distinct primes. Here each prime $e_i$ is uniquely determined by the $i$-bit prefix $M_{|i}$ of the message $M$, and for each generation of $e_i$ a number of primality tests have to be executed which is the dominant running time of signing (and verifying). The above signature scheme is only weakly secure under the RSA

[1] There are exceptions, e.g., by Waters [53] (CDH assumption in bilinear groups), Hohenberger and Waters [40], and the lattice-based schemes [18,14] (SIS assumption). However, these are not among the most efficient "Hash-and-sign"-type schemes.

[2] We stress that the PHF parameter $m$ does *not* directly correspond to the number of signatures that can be created during the security reduction. Rather, $m$ indicates how many collisions of (honestly generated) $e$-values we can handle in the reduction. Hence, the larger $m$ is, the smaller $e$ can be chosen.

assumption, and a chameleon hash has to be used to make it fully secure, thereby doubling the signature size to two elements from $\mathbb{Z}_N$ and adding $\approx$ 2kbit to the public-key size [39]. The main disadvantage of HW signatures is, however, the generation and testing of the $|M|$ primes $e_1, \ldots, e_{|M|}$ necessary to compute the hash function $\mathsf{P}(M)$. Concretely, for $k = 80$ bits security, HW signatures need to generate $|M| = 160$ random primes for the signing process.

### 1.1 Summary of Our Contributions

As the main technical contribution we propose several new constructions of $(m, 1)$-PHFs for any $m \geq 1$. In particular, we solve the open problem posed in [38] of constructing deterministic $(m, 1)$-PHFs for $m > 2$. Even though our main applications are digital signatures we remark that PHFs are a very general framework for designing and analyzing cryptographic protocols in the Diffie-Hellman and RSA setting. For example, in [38], it was shown that PHFs imply collision-resistant hash functions and lead to elegant and simple proofs of Waters' IBE and signature schemes [53] and its countless variants (e.g., [15,7]). More importantly, a large body of cryptographic protocols with security in the standard model are using — implicitly or explicitly — the partitioning trick that is formalized in PHFs. To mention only a few examples, this ranges from collision-resistant hashing [20,4], digital signature schemes [12,53] (also in various flavors [47,51,8]), chosen-ciphertext secure encryption [15,41,35,37,14], identity-based encryption [9,10,42,18,1], attribute-based encryption [49] to symmetric authentication [43]. We expect that our new PHF constructions can also be applied to some of the mentioned applications.

We also show how to use our new $(m, 1)$-PHFs for generic constructions of short yet efficient hash-and-sign signatures whose security is based on weaker hardness assumptions: the $q$-DH and the RSA assumption. Whereas our $q$-DH schemes $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}]$ are (to the best of our knowledge) the first hash-and-sign schemes from this assumption, our RSA schemes $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}]$ and $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}]$ are conceptually different from HW signatures and we obtain a considerable efficiency improvement. A large number of new signature schemes with different tradeoffs can be derived by combining the generic signature schemes with PRFs. An overview of the efficiency of some resulting schemes and a comparison with existing schemes from [23,29,11,38,40] is provided in Table 1. Our new schemes offer different tradeoffs between signature size, efficiency, and public-key size. The bigger the parameter $m$ in the $(m, 1)$-PHF, the larger the public-key size, the shorter the signatures. To obtain extremely short and/or efficient signatures, the size of the public key can get quite large. Concretely, with a public-key of size 26mbit we obtain 200 bit signatures from the (Strong) $q$-DH assumption. These are the shortest knwon standard-model digital signatures in bilinear groups. Remarkably, $\mathsf{Sig}_{\mathsf{SRSA}}[\mathsf{H}_{\mathsf{cfs}}]$ which instatiates the Strong RSA signatures from [38] with our new $(m, 1)$-PHF $\mathsf{H}_{\mathsf{cfs}}$ for $m \geq 6$, results in a hash-and-sign signature scheme where the signing procedure is dominated by one single modular exponentiation. This is the first RSA-based signature scheme whose signing

complexity is not dominated by generating random primes.[3] Hence signing is essentially as efficient as RSA full-domain-hash [6] with the drawback of a huge public-key.

While these short signatures are mostly of theoretical interest and contribute to the problem of determining concrete bounds on the size of standard-model signatures, we think that in certain applications even a large public-key is tolerable. In particular, our public key sizes are still comparable to the ones of recently proposed lattice-based signatures [46,30,18,14].

We note furthermore, that it is possible to apply efficiency improvements from [40] to our RSA-based schemes as well. This allows us to reduce the number of primality tests required for signing and verification sigificantly. More precisely, it is possible to transform each signature scheme requiring $\lambda$ primality tests into a scheme which requires only $\lambda/c$ primality tests, at the cost of loosing a factor of $2^{-c}$ in the security reduction. For example, $\mathsf{Sig}^{*}_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{Weak}}]^{\S}$ with $m = 11$ and $c = 40$ is a RSA-based signature scheme which requires only a *single* primality test for signing and verification, at the cost of loosing a factor of $2^{-40}$ in the security reduction.

### 1.2 Details of Our Contributions

Our main technical contribution to obtain shorter signatures are several new constructions of $(m, 1)$-PHFs for $m \geq 2$ (cf. Table 2 in Section 3). Using cover-free sets, we construct a deterministic $(m, 1)$-PHF $\mathsf{H}_{\mathsf{cfs}}$ with public parameters of $O(km^2)$ group elements. This solves the problem from [38] of constructing deterministic $(m, 1)$-PHFs for $m > 2$. We remark that cover-free sets were already used in [25,33,22] to construct identity-based encryption schemes. Furthermore, we propose a randomized $(m, 1)$-PHF $\mathsf{H}_{\mathsf{rand}}$ with public parameters of $O(m^2)$ group elements and small randomness space. Finally, we construct a weakly secure deterministic $(m, 1)$-PHF $\mathsf{H}_{\mathsf{Weak}}$ with public parameters of $m$ group elements. The latter PHF already appeared implicitly in the context of identity/attribute-based encryption [19,49] (generalizing [9]). Weakly secure PHFs only yield weakly secure signature schemes that need to be "upgraded" to fully secure schemes using a chameleon hash function.

RSA Signatures. Our new RSA signatures $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}]$ are of the form

$$\mathsf{sig}(M) = (\mathsf{H}(M)^{1/\mathsf{P}(s)} \bmod N, s), \tag{1}$$

where $s$ is a short random bitstring, $\mathsf{H}(\cdot)$ is a $(m, 1)$-PHF, and $\mathsf{P}(s) := e_1 \cdot \ldots \cdot e_{|s|}$ is the product of $|s|$ primes $e_1, \ldots, e_{|s|}$, where the $i$th prime is uniquely determined by the $i$th prefix $s_{|i}$ of the randomness $s$. (If the PHF $\mathsf{H}$ is probabilistic, $\mathsf{sig}$ additionally contains a small random bitstring $r$.) Our security proof is along the lines of [38], but

[3] Since the complexity of finding a random $\mu$-bit prime with error $2^{-k}$ is $O(k\mu^4)$, we expect that for $\mu \approx 60$ (or, equivalently, using $\mathsf{H}_{\mathsf{cfs}}$ with $m \geq 6$) a full exponentiation modulo a 1024-bit integer become roughly as expensive as generating a random $\mu$-bit prime.

**Table 1.** Signature sizes of different schemes. Rows with grey background indicate new results from this paper. The chosen parameters provide unforgeability with $k = 80$ bits of security after revealing maximally $q = 2^{30}$ signatures. RSA signatures are instantiated with a modulus of $|N| = 1024$ bits, Bilinear signatures in asymmetric pairings using a BN curve [3] with $\log p = 160$ bits. (In this, we actually ignore the *multiplicative* reduction loss between a forger and, e.g., an RSA adversary.) We assume that elements in $\mathbb{G}_1$ can be represented by $|\mathbb{G}_1| = 160$ bits, while an element $\mathbb{G}_2$ by $|\mathbb{G}_2| = 320$ bits. The description of the bilinear group/modulus $N$ is not counted in the public key. We assume $2k = 160$-bit messages in order to provide $k$=80 bits of security (to sign longer messages, we can apply a collision-resistant hash function first). The efficiency column counts the dominant operations for signing. For Bilinear and RSA signatures this counts the number of modular exponentiations, for RSA signatures $k \times \mathsf{P}_\mu$ counts the number of random $\mu$-bit primes that need to be generated to evaluate function $\mathsf{P}(\cdot)$. (For $\mu \gg 60$, $1 \times \mathsf{P}_\mu$ takes more time than $1 \times \mathsf{Exp}$.) *The RSA-based chameleon hash function from [39] (which builds upon [2]) was used (adding $1 \times |\mathbb{Z}_N|$ to signature size). §Security reduction loses an additional factor of $2^{40}$.

| Signature scheme | | Assumption | Sig. Size | Efficiency | PK size |
|---|---|---|---|---|---|
| Waters [53] | | CDH | 320 | $2 \times \mathsf{Exp}$ | 26k |
| Boneh-Boyen [11] | | Strong $q$-DH | 320 | $1 \times \mathsf{Exp}$ | 640 |
| $\mathsf{Sig}_{\mathsf{S}\text{-}q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{Wat}}]$ [38] | | Strong $q$-DH | 230 | $1 \times \mathsf{Exp}$ | 26k |
| $\mathsf{Sig}_{\mathsf{S}\text{-}q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}]$ | $(m{=}8)$ | Strong $q$-DH | 200 | $1 \times \mathsf{Exp}$ | 26m |
| $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{Wat}}, \mathsf{H}_{\mathsf{Wat}}]$ | $(m{=}2)$ | $q$-DH | 230 | $1 \times \mathsf{Exp}$ | 48k |
| $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}, \mathsf{H}_{\mathsf{Wat}}]$ | $(m{=}8)$ | $q$-DH | 200 | $1 \times \mathsf{Exp}$ | 26m |
| Cramer-Shoup [23] | | Strong RSA | 2208 | $1 \times \mathsf{P}_{160}$ | 3k |
| Gennaro et. al.* [29] | | Strong RSA | 2048 | $1 \times \mathsf{P}_{160}$ | 3k |
| $\mathsf{Sig}_{\mathsf{SRSA}}[\mathsf{H}_{\mathsf{Wat}}]$ [38] | | Strong RSA | 1104 | $1 \times \mathsf{P}_{80}$ | 128k |
| $\mathsf{Sig}_{\mathsf{SRSA}}[\mathsf{H}_{\mathsf{cfs}}]$ | $(m{=}6)$ | Strong RSA | 1068 | $\approx 1 \times \mathsf{Exp}$ | 94m |
| $\mathsf{Sig}^*_{\mathsf{SRSA}}[\mathsf{H}_{\mathsf{Weak}}]$ | $(m{=}6)$ | Strong RSA | 2092 | $\approx 2 \times \mathsf{Exp}$ | 9k |
| Hohenberger-Waters* [40] | | RSA | 2048 | $160 \times \mathsf{P}_{1024}$ | 3k |
| $\mathsf{Sig}^*_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{Weak}}]$ | $(m{=}2)$ | RSA | 2048 | $70 \times \mathsf{P}_{1024}$ | 5k |
| $\mathsf{Sig}^*_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{Weak}}]$ | $(m{=}4)$ | RSA | 2048 | $50 \times \mathsf{P}_{1024}$ | 7k |
| $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{Wat}}]$ | $(m{=}2)$ | RSA | 1094 | $70 \times \mathsf{P}_{1024}$ | 128k |
| $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{rand}}]$ | $(m{=}4)$ | RSA | 1214 | $50 \times \mathsf{P}_{1024}$ | 32k |
| $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{cfs}}]$ | $(m{=}4)$ | RSA | 1074 | $50 \times \mathsf{P}_{1024}$ | 40m |
| $\mathsf{Sig}^*_{\mathsf{RSA}}[\mathsf{H}_{\mathsf{Weak}}]^{\S}$ | $(m{=}11)$ | RSA | 2048 | $1 \times \mathsf{P}_{1024}$ | 14k |

using $\mathsf{P}$ enables a reduction to the RSA assumption (Theorem 7) in the standard model. The main conceptual novelty is that we apply $\mathsf{P}$ to the randomness $s$ rather than the message $M$ as in HW signatures. Because the values $s$ are relatively small, our scheme is considerably more efficient than that of [40].

Concretely, the length of $s$ is controlled by the PHF parameter $m$ as $|s| = \log q + k/m$, where $q$ is an upper bound on the number of signatures the scheme supports. (See the full version [34] for a formal argument.) For $k = 80$ bits security and $q = 2^{30}$ (as recommended in [6]) we can make use of our new constructions of $(m, 1)$-PHFs with $m \geq 2$. For example, with a $(4, 1)$-PHF, the bitstring $s$ can be as small as 50 bits which leads to very small signatures. More

importantly, since the function $\mathsf{P}(s)$ only has to generate $|s|$ distinct primes $e_1, \ldots, e_{|s|}$ (compared to $|M| \gg |s|$ primes in HW signatures), the signing and verification algorithms are considerably faster. The drawback of our new signature scheme is that the system parameters of $\mathsf{H}$ grow with $m$.

Bilinear Signatures. Our new $q$-DH signatures $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}]$ are of the form

$$\mathsf{sig}(M) = (\mathsf{H}(M)^{1/\mathsf{d}(s)}, s), \tag{2}$$

where again $s$ is a short random bitstring, $\mathsf{H}$ is a $(m, 1)$ programmable hash function, and $\mathsf{d}(\cdot)$ is a special (secret) function mapping bitstrings to $\mathbb{Z}_p$. Since $\mathsf{D}(s) := g^{\mathsf{d}(s)}$ can be computed publicly, verification is done by using the properties of the bilinear group. Security is proved under the $q$-DH assumption in the standard model. Similar to our RSA-based signatures the length of $s$ is controlled by the PHF parameter $m$. For example, for $m = 8$ we obtain standard-model signatures of size $|\mathbb{G}| + |s| = 160 + 40 = 200$ bits. We have to refer to the full version [34] for details.

Full-Domain Hash Signatures. We remark that full-domain hash signature schemes over a homomorphic domain (e.g., RSA-FDH [6] and BLS signatures [13]) instantiated with $(m, 1)$-PHFs provide efficient $m$-time signature schemes without random oracles. This nicely complements the impossibility results from [26] who show that without the homomorphic property this is not possible. We remark that an instantiation of RSA-FDH as a $m$-time signature scheme was independently observed in [24].

Proof Techniques and Related Work. Our RSA-based signature scheme represents a combination of techniques from [38] and [40]. Namely, in the basic RSA-based signature scheme from [38], a signature is of the form $(\mathsf{H}(M)^{1/s} \bmod N, s)$ for a prime $s$. The use of a *programmable* hash function $\mathsf{H}$ enables very efficient schemes, whose security however cannot be reduced to the standard (non-strong) RSA problem, since a forged signature $(\mathsf{H}(M)^{1/s^*}, s^*)$ corresponds to an RSA inversion with adversarially chosen exponent $s^*$. On the other hand, the (basic, weakly secure) signature scheme from [40] is of the form $g^{1/\mathsf{P}(M)} \bmod N$. The special structure of $\mathsf{P}$ (which maps a message $M$ to the product of $|M|$ primes) makes it possible to prove security under the standard RSA assumption. However, since $\mathsf{P}$ is applied to messages (i.e., 160-bit strings), evaluation of $\mathsf{P}$ requires a large number of primality tests. We combine the best of both worlds with signatures of the form $(\mathsf{H}(M)^{1/\mathsf{P}(s)} \bmod N, s)$ for *short* (e.g., 40-bit) random strings $s$. In contrast to the scheme of [40], this directly yields a fully secure signature scheme, so we do not need a chameleon hash function.

In the security proof of our RSA signatures we distinguish between two types of forgers: type I forgers recycle a value from $\{s_1, \ldots, s_q\}$ for the forgery, where the $s_i$'s are the random bitstrings used for the simulated signatures; type II forgers use a new value $s^* \notin \{s_1, \ldots, s_q\}$ for the forgery and therefore are more difficult to reduce to the RSA assumption. For the reduction of type II forgers to the RSA assumption we can use a clever "prefix-guessing" technique from [40]

to embed the prime $e$ from the RSA challenge in the function $\mathsf{P}(\cdot)$ such that the product $\mathsf{P}(s^*)$ contains $e$.[4] Similar to the proof of HK signatures [38], the reduction for Type I forgers makes use of the $(m, 1)$ programmability of $\mathsf{H}(\cdot)$.

Strong $q$-DH signatures from [38] can actually be viewed as our $q$-DH signatures from (2) instantiated with the special function $\mathsf{d}(s) = x+s$ (where $x$ is part of the secret-key). In our scheme, the leverage to obtain security from $q$-DH is that the function $\mathsf{D}(s) := g^{\mathsf{d}(s)}$ acts as a $(\mathsf{poly}, 1)$-PHF. That is, $\mathsf{d}(\cdot)$ can be setup such that (with non-negligible probability) $\mathsf{d}(s_i) = x + a(s_i)$ for $a(s_i) \neq 0$ but $\mathsf{d}(s^*) = x$, where $s_1, \ldots, s_q$ is the randomness used for the generated signatures and $s^*$ is the randomness used for the forgery.

### 1.3 Open Problems

A number of interesting open problems remain. We ask how to construct (deterministic) $(m, 1)$-PHFs for $m \geq 1$ with smaller parameters than the ones from Table 2. Since the constructions of cover free sets are known to be optimal up to a log factor, a new method will be required. Furthermore, obtaining truely practical signatures from the RSA or factoring assumption is still an open problem. In particular, we ask for a construction of hash-and-sign (strong) RSA signatures that do not require the generation of primes at signing.

## 2 Preliminaries

For $k \in \mathbb{N}$, we write $1^k$ for the string of $k$ ones, and $[k]$ for $\{1, \ldots, k\}$. Moreover, $|x|$ denotes the length of a bitstring $x$, while $|S|$ denotes the size of a set $S$. Further, $s \xleftarrow{\$} S$ denotes the sampling a uniformly random element $s$ of $S$. For an algorithm $\mathcal{A}$, we write $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is a (probabilistic) algorithm that outputs $z$ on input $(x, y, \ldots)$.

### 2.1 Digital Signatures

A digital signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vfy})$ consists of three algorithms. Key generation $\mathsf{Gen}$ generates a keypair $(pk, sk) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for a secret signing key $sk$ and a public verification key $pk$. The signing algorithm $\mathsf{Sign}$ inputs a message and the secret signing key, and returns a signature $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$ of the message. The verification algorithm $\mathsf{Vfy}$ takes a verification key and a message with corresponding signature as input, and returns $b \leftarrow \mathsf{Vfy}(pk, m, \sigma)$ where $b \in \{\texttt{accept}, \texttt{reject}\}$. We require the usual correctness properties.

[4] More precisely, when simulating a type II forger, the values $s_1, \ldots, s_q$ are known in advance to the simulator. Since $s^* \notin \{s_1, \ldots, s_q\}$ there is some prefix $s^*_{|i}$ of $s^*$ that is different from all prefixes of $s_1, \ldots, s_q$. We can guess the smallest such prefix such that the simulator knows $s^*_{|i}$ from the forgery at the beginning. This knowledge can be used to embed $e$ from the RSA challenge in the function $\mathsf{P}(\cdot)$ such that the product $\mathsf{P}(s^*)$ contains $e$.

Let us recall the *existential unforgeability against chosen message attacks* (EUF-CMA) security experiment [31], played between a challenger and a forger $\mathcal{F}$.

1. The challenger runs $\mathsf{Gen}$ to generate a keypair $(pk, sk)$. The forger receives $pk$ as input.
2. The forger may ask the challenger to sign a number of messages. To query the $i$-th signature, $\mathcal{F}$ submits a message $m_i$ to the challenger. The challenger returns a signature $\sigma_i$ under $sk$ for this message.
3. The forger outputs a message $m^*$ and signature $\sigma^*$.

$\mathcal{F}$ wins the game, if $\mathtt{accept} \leftarrow \mathsf{Vfy}(pk, m^*, \sigma^*)$, that is, $\sigma^*$ is a valid signature for $m^*$, and $m^* \neq m_i$ for all $i$. We say that $\mathcal{F}$ $(t, q, \epsilon)$-breaks the EUF-CMA security of $\mathsf{Sig}$, if $\mathcal{F}$ runs in time $t$, makes at most $q$ signing queries, and has success probability $\epsilon$. We say that $\mathsf{Sig}$ is EUF-CMA secure, or $\mathsf{Sig}$ is *fully* secure, if $\epsilon$ is negligible for any probabilistic polynomial-time algorithm $\mathcal{F}$.

We also say, that a scheme is *weakly* secure, if it meets the above security definition, but the adversary can not choose the messages to be signed adaptively. Instead it has to commit to a list $m_1, \ldots, m_q$ before seeing the public key. There exist efficient generic techniques to convert a weakly secure signature scheme into a fully secure one, e.g., using chameleon hashes [44].

### 2.2 Prime Numbers, Factoring, and the RSA Assumption

For $x \in \mathbb{N}$ let $\pi(x)$ denote the number of primes between 0 and $x$. The following lemma is a direct consequence of Chebyshev's bounds on $\pi(x)$ (see [32], for instance).

**Lemma 1.** $\frac{x}{\log_2 x} < \pi(x) < \frac{2x}{\log_2 x}$

We say that a prime $p$ is a *safe* prime, if $p = 2p' + 1$ and $p'$ is also prime. Let $p$ and $q$ be two randomly chosen $k/2$-bit safe primes, and let $N = pq$. Let $e \in \mathbb{Z}_{\phi(n)}$ be a random integer, relatively prime to $\phi(N)$. We say that an algorithm $\mathcal{A}$ $(t, \epsilon)$-breaks the RSA assumption, if $\mathcal{A}$ runs in time $t$ and

$$\Pr[y^{1/e} \xleftarrow{\$} \mathcal{A}(N, e, y)] \geq \epsilon.$$

We assume that there exists no algorithm that $(t, \epsilon)$-breaks the RSA assumption with polynomial $t$ and non-negligible $\epsilon$.

We denote with $\mathsf{QR}_N$ the group of quadratic residues modulo $N$. The following lemma, which is due to Shamir [52], is useful for the security proof of the generic RSA-based signature scheme described in Section 4.

**Lemma 2.** *There is an efficient algorithm that, on input $y, z \in \mathbb{Z}_N$ and integers $e, f \in \mathbb{Z}$ such that $\gcd(e, f) = 1$ and $z^e \equiv y^f \bmod n$, computes $x \in \mathbb{Z}_N$ satisfying $x^e \equiv y \bmod N$.*

### 2.3 Generalized Birthday Bound

Although not explicitly stated, the following lemma is implicit in [36]. We will apply it several times in the security proofs for our generic signature schemes.

**Lemma 3.** *Let $A$ be a set with $|A| = a$. Let $X_1, \ldots, X_q$ be $q$ independent random variables, taking uniformly random values from $A$. Then the probability that there exist $m+1$ pairwise distinct indices $i_1, \ldots, i_{m+1}$ such that $X_{i_1} = \cdots = X_{i_{m+1}}$ is upper bounded by $\frac{q^{m+1}}{a^m}$.*

## 3 Programmable Hash Functions

### 3.1 Definitions

Let $G = (\mathbb{G}_k)$ be a family of groups, indexed by security parameter $k \in \mathbb{N}$. We omit the subscript when the reference to the security parameter is clear, thus write $\mathbb{G}$ for $\mathbb{G}_k$.

A *group hash function* $H$ over $\mathbb{G}$ with input length $l = l(k)$ consists of two efficient algorithms $\mathsf{PHF.Gen}$ and $\mathsf{PHF.Eval}$. The probabilistic algorithm $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$ generates a hash key $\kappa$ for security parameter $k$. Algorithm $\mathsf{PHF.Eval}$ is a deterministic algorithm, taking as input a hash function key $\kappa$ and $X \in \{0,1\}^l$, and returning $\mathsf{PHF.Eval}(\kappa, X) \in \mathbb{G}$.

**Definition 1.** *We say that a group hash function $H = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ is* $(m, n, \gamma, \delta)$-programmable*, if there is an efficient trapdoor generation algorithm $\mathsf{PHF.TrapGen}$ and an efficient trapdoor evaluation algorithm $\mathsf{PHF.TrapEval}$ with the following properties.*

1. *The probabilistic algorithm $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ takes as input group elements $g, h \in \mathbb{G}$, and produces a hash function key $\kappa$ together with trapdoor information $\tau$.*
2. *For all generators $g, h \in \mathbb{G}$, the keys $\kappa, \kappa'$, where $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$ and $\kappa' \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$, are statistically $\gamma$-close.*
3. *On input $X \in \{0,1\}^l$ and trapdoor information $\tau$, the deterministic trapdoor evaluation algorithm $(a_X, b_X) \leftarrow \mathsf{PHF.TrapEval}(\tau, X)$ produces $a_X, b_X \in \mathbb{Z}$ so that for all $X \in \{0,1\}^l$,*

$$\mathsf{PHF.Eval}(\kappa, X) = g^{a_X} h^{b_X}$$

4. *For all $g, h \in \mathbb{G}$, all $\kappa$ generated by $\kappa \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$, and all $X_1, \ldots, X_m \in \{0,1\}^l$ and $Z_1, \ldots, Z_n \in \{0,1\}^l$ such that $X_i \neq Z_j$ for all $i, j$, we have*

$$\Pr[a_{X_1} = \cdots = a_{X_m} = 0 \text{ and } a_{Z_1}, \ldots, a_{Z_n} \neq 0] \geq \delta,$$

*where $(a_{X_i}, b_{X_i}) = \mathsf{PHF.TrapEval}(\tau, X_i)$, $(a_{Z_j}, b_{Z_j}) = \mathsf{PHF.TrapEval}(\tau, Z_j)$, and the probability is taken over the trapdoor $\tau$ produced along with $\kappa$.*

*We also say that $H$ is* $(m, n)$-programmable *for short, if $\gamma$ is negligible and $\delta$ is noticeable. If $H$ is $(1, q)$-programmable for every polynomial $q = q(k)$, then we say that $H$ is* $(1, \mathsf{poly})$*-programmable.*

In settings in which the group order is hidden, we will use a refinement of the PHF definition:

**Definition 2.** *A group hash function* $H = (\mathsf{RPHF.Gen}, \mathsf{RPHF.Eval})$ *is* evasively $(m, n, \gamma, \delta)$-programmable*, if it is* $(m, n, \gamma, \delta)$*-programmable as in Definition 1, but with the strengthened requirement*

*4'. For* all prime numbers $e$ with $2^l < e \leq |\mathbb{G}|$*, all* $g, h \in \mathbb{G}$*, and all* $\kappa$ *generated by* $\kappa \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$*, and all* $X_1, \ldots, X_m \in \{0,1\}^l$ *and* $Z_1, \ldots, Z_n \in \{0,1\}^l$ *such that* $X_i \neq Z_j$ *for all* $i, j$*, we have*

$$\Pr[a_{X_1} = \cdots = a_{X_m} = 0 \text{ and } \gcd(a_{Z_1}, e) = \cdots = \gcd(a_{Z_n}, e) = 1] \geq \delta.$$

*Here* $a_{X_i}$ *and* $a_{Z_j}$ *denote the output of the trapdoor evaluation algorithm* $(a_{X_i}, b_{X_i}) = \mathsf{PHF.TrapEval}(\tau, X_i)$ *and* $(a_{Z_j}, b_{Z_j}) = \mathsf{PHF.TrapEval}(\tau, Z_j)$*, and the probability is taken over the trapdoor* $\tau$ *produced along with* $\kappa$*.*

Hofheinz and Kiltz [36] have also introduced the notion of *randomized* programmable hash functions. A *randomized* group hash function $H$ with input length $l = l(k)$ and randomness space $R = (\mathcal{R}_k)$ consists of two efficient algorithms $\mathsf{RPHF.Gen}$ and $\mathsf{RPHF.Eval}$. Algorithm $\mathsf{RPHF.Gen}$ is probabilistic, and generates a hash key $\kappa \xleftarrow{\$} \mathsf{RPHF.Gen}(1^k)$ for security parameter $k$. The deterministic algorithm $\mathsf{RPHF.Eval}$ takes randomness $r \in \mathcal{R}_k$ and $X \in \{0,1\}^l$ as input, and returns a group element $\mathsf{RPHF.Eval}(\kappa, X) \in \mathbb{G}$.

**Definition 3.** *Let* $H = (\mathsf{RPHF.Gen}, \mathsf{RPHF.Eval})$ *be a randomized group hash function. We say that* $H$ *is* $(m, n, \gamma, \delta)$*-programmable, if there are efficient algorithms* $\mathsf{RPHF.TrapGen}$*,* $\mathsf{RPHF.TrapEval}$*, and* $\mathsf{RPHF.TrapRand}$ *such that:*

1. *The probabilistic algorithm* $\mathsf{RPHF.TrapGen}(1^k, g, h)$ *takes as input group elements* $g, h \in \mathbb{G}$*, and produces a key* $\kappa$ *and trapdoor* $\tau$*. For all generators* $g, h \in \mathbb{G}$*, the keys* $\kappa \xleftarrow{\$} \mathsf{RPHF.Gen}(1^k)$ *and* $\kappa' \xleftarrow{\$} \mathsf{RPHF.TrapGen}(1^k, g, h)$ *are statistically* $\gamma$*-close.*
2. *The deterministic trapdoor evaluation algorithm takes as input* $X \in \{0,1\}^l$ *and* $r \in \mathcal{R}_k$*, and produces two functions* $(a_X(\cdot), b_X(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(\tau, X, r)$ *such that for all* $X \in \{0,1\}^l$*,*

$$\mathsf{RPHF.Eval}(\kappa, X, r) = g^{a_X(r)} h^{b_X(r)}.$$

3. *On input of trapdoor* $\tau$*,* $X \in \{0,1\}^l$*, and index* $i \in [m]$*, the* $\mathsf{RPHF.TrapRand}$ *algorithm produces* $r \leftarrow \mathsf{RPHF.TrapRand}(\tau, X, i)$ *with* $r \in \mathcal{R}_k$*. For all* $g, h \in \mathbb{G}$*, all* $\kappa$ *generated by* $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$*, all* $X_1, \ldots, X_m$*, and* $r_{X_i} = \mathsf{RPHF.TrapRand}(\tau, X_i, i)$*, we require that the* $r_{X_i}$ *are independent and uniformly distributed random variables over* $\mathcal{R}_k$*.*
4. *For all* $g, h \in \mathbb{G}$ *and all* $\kappa$ *generated by* $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$*, all* $X_1, \ldots, X_m \in \{0,1\}^l$ *and* $Z_1, \ldots, Z_n \in \{0,1\}^l$ *such that* $X_i \neq Z_j$*, and for all* $\tilde{r}_1, \ldots, \tilde{r}_n \in \mathcal{R}_k$ *and* $r_{X_i} \leftarrow \mathsf{RPHF.TrapRand}(\tau, X_i, i)$*, we have*

$$\Pr[a_{X_1}(r_{X_1}) = \cdots = a_{X_m}(r_{X_m}) = 0 \text{ and } a_{Z_1}(\tilde{r_1}), \ldots, a_{Z_n}(\tilde{r_n}) \neq 0] \geq \delta,$$

*where the* $a_{X_i}$ *and* $a_{Z_j}$ *are the output of the trapdoor evaluation* $(a_{X_i}, b_{X_i}) = \mathsf{RPHF.TrapEval}(\tau, X_i, r_{X_i})$ *and* $(a_{Z_j}, b_{Z_j}) = \mathsf{PHF.TrapEval}(\tau, Z_j, \tilde{r}_j)$*, and the*

**Table 2.** Overview of our constructions of (randomized/weak) programmable hash functions. Rows with grey background are new constructions from this paper.

| Name | Type | Param. $(m, n)$ | Size of $\kappa$ | Randomness |
|---|---|---|---|---|
| $\mathsf{H}_{\mathsf{Wat}}$ [53,36] (§ 3.2) | PHF | $(1, \mathsf{poly})$ and $(2, 1)$ | $(l+1) \times \lvert\mathbb{G}\rvert$ | — |
| $\mathsf{H}_{\mathsf{cfs}}$ (§ 3.3) | PHF | $(m, 1)$ | $(16m^2l + 1) \times \lvert\mathbb{G}\rvert$ | — |
| $\mathsf{H}_{\mathsf{rand}}$ (§ 3.4) | RPHF | $(m, 1)$ | $(2m^2 + 1) \times \lvert\mathbb{G}\rvert$ | $\{0,1\}^l$ |
| $\mathsf{H}_{\mathsf{Weak}}$ (§ 3.5) | weak PHF | $(m, 1)$ | $(m+1) \times \lvert\mathbb{G}\rvert$ | — |

*probability is taken over the trapdoor $\tau$ produced along with $\kappa$. Here $X_i$ may depend on $X_j$ and $r_{X_j}$ for $j < i$, and the $Z_1, \ldots, Z_n$ may depend on all $X_i$ and $r_i$.*

*Again we omit $\gamma$ and $\delta$, if $\gamma$ is negligible and $\delta$ is noticeable.* Randomized evasively programmable hash functions *are defined as in Definition 2.*

In the remainder of this Section we propose a number of new PHFs offering different trade-offs. Our results are summarized in Table 2.

### 3.2 Multi-generator Programmable Hash Function

The programmable hash function described in Definition 4 below was (implicitly) introduced in [53]. An explicit analysis can be found in [36].

**Definition 4.** *Let $G = (\mathbb{G}_k)$ be a group family, and $l = l(k)$ be a polynomial. Let $\mathsf{H}_{\mathsf{Wat}} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ be defined as follows.*

- $\mathsf{PHF.Gen}(1^k)$ *returns $\kappa = (h_0, \ldots, h_l)$, where $h_i \xleftarrow{\$} \mathbb{G}_k$ for $i \in [l]$.*
- *On input $X = (x_1, \ldots, x_l) \in \{0,1\}^l$ and $\kappa = (h_0, \ldots, h_l)$, $\mathsf{PHF.Eval}(\kappa, X)$ returns*

$$\mathsf{PHF.Eval}(\kappa, X) = h_0 \prod_{i=1}^{l} h_i^{x_i}.$$

**Theorem 1 (Theorem 3.6 of [36]).** *For any fixed polynomial $q = q(k)$ and any group with known order, $\mathsf{H}_{\mathsf{Wat}}$ is evasively $(1, q, 0, O(1/(q\sqrt{l})))$-programmable and $(2, 1, 0, O(1/l))$-programmable hash function.*

Although evasive programmability was not introduced in [36], it follows from their proof, since the values of $a_{Z_j}$ that occur there are bounded in the sense $|a_{Z_j}| < 2^l$. We remark that Theorem 1 also carries over to groups of unknown order.

### 3.3 A New Deterministic Programmable Hash Function

Let $S, T$ be sets. We say that $S$ does not cover $T$, if $T \not\subseteq S$. Let $d, m, s$ be integers, and let $F = (F_i)_{i \in [s]}$ be a family of $s$ subsets of $[d]$. We say that $F$ is $m$-cover free, if for any set $I$ containing (up to) $m$ indices $I = \{i_1, \ldots, i_m\} \subseteq [s]$,

it holds that $F_j \not\subseteq \bigcup_{i\in I} F_i$ for any $j$ which is *not* contained in $I$. In other words, if $|I| \leq m$, then the union $\bigcup_{i\in I} F_i$ is not covering $F_j$ for all $j \in [s] \setminus I$. We say that $F$ is $w$-uniform, if $|F_i| = w$ for all $i \in [s]$.

**Lemma 4 ([27,45]).** *There is a deterministic polynomial-time algorithm that, on input of integers $m, s = 2^l$, returns $d \in \mathbb{N}$ and set family $F = (F_i)_{i\in[s]}$ such that $F$ is $m$-cover free over $[d]$ and $w$-uniform, where $d \leq 16m^2 l$ and $w = d/4m$.*

In the following we will associate $X \in \{0,1\}^l$ to a subset $F_i$, $i \in [s]$, by interpreting $X$ as an integer in the range $[0, 2^l - 1]$, and setting $i = X + 1$. We will write $F_X$ to denote the subset associated to $X$.

**Definition 5.** *Let $G = (\mathbb{G}_k)$ be a group family, and $l = l(k)$ and $m = m(k)$ be polynomials. Let $s = 2^l$, $d = 16m^2 l$, and $w = d/4m$. We define a hash function $\mathsf{H}_{\mathsf{cfs}} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ be as follows.*

- $\mathsf{PHF.Gen}(1^k)$ *returns $\kappa = (h_1, \ldots, h_d)$, where $h_i \xleftarrow{\$} \mathbb{G}_k$ for $1 \leq i \leq d$.*
- *Let $F_X \subseteq [d]$ be the subset associated to $X \in [0, 2^l - 1]$. On input $X$ and $\kappa = (h_1, \ldots, h_d)$, $\mathsf{PHF.Eval}(\kappa, X)$ returns*

$$\mathsf{PHF.Eval}(\kappa, X) = \prod_{i\in F_X} h_i.$$

**Theorem 2.** *Let $\mathbb{G} = \mathbb{G}_k$ be a group of known order $p$. $\mathsf{H}_{\mathsf{cfs}}$ is an evasively $(m, 1, \gamma, \delta)$-programmable hash function with $\gamma = 0$ and $\delta = 1/(16m^2 l)$.*

*Proof.* Consider the following algorithms.

- $\mathsf{PHF.TrapGen}(1^k, g, h)$ samples $d$ uniformly random integers $b_1, \ldots, b_d \xleftarrow{\$} \mathbb{Z}_p$ and an index $t \xleftarrow{\$} [d]$. Then it sets $h_t = gh^{b_t}$, and $h_i = h^{b_i}$ for all $i \in [1, d]$ with $i \neq t$. $\mathsf{PHF.TrapGen}$ returns $(\kappa, \tau)$ with $\tau = (t, b_1, \ldots, b_d)$ and $\kappa = (h_1, \ldots, h_d)$.
- On input $(\tau, X)$, $\mathsf{PHF.TrapEval}$ sets $b_X = \sum_{i\in F_X} b_i$, and $a_X = 1$ if $t \in F_X$, and $a_X = 0$ if $t \notin F_X$, and returns $(a_X, b_X)$.

$\mathsf{PHF.TrapGen}$ outputs a vector of independent and uniformly distributed group elements, thus we have $\gamma = 0$. Fix $X_1, \ldots, X_m, Z \in [0, 2^l - 1]$. Since $F$ is a $m$-cover free set family, there must be an index $t'$ such that $t' \notin \bigcup_{i=1}^m F_{X_i}$, but $t' \in F_Z$. Since $t$ is picked uniformly random among $16m^2 l$ possibilities, we have $t = t'$, and thus $a_{X_i} = 0$ and $a_Z = 1$, with probability $\delta = 1/(16m^2 l)$. Finally, $a_Z = 1$ implies $\gcd(a_Z, e) = 1$ for all primes $e$, thus $\mathsf{H}_{\mathsf{cfs}}$ is evasively programmable.

Theorem 2 can be generalized to groups of hidden order. The proof proceeds exactly like the proof of Theorem 2, except that we have to approximate the group order. E.g., for the group of quadratic residues $\mathsf{QR}_n$, we can sample random exponents $b_i \xleftarrow{\$} \mathbb{Z}_{n^2}$. This way, we can sample nearly uniform ($1/\sqrt{n}$-close) group elements $h_i = h^{b_i}$, which yields the following theorem.

**Theorem 3.** *Let $\mathbb{G} = \mathsf{QR}_n$ be the group of quadratic residues modulo $n = pq$, where $p$ and $q$ are safe distinct primes. $\mathsf{H}_{\mathsf{cfs}}$ is a $(m, 1, \gamma, \delta)$-evasively programmable hash function over $\mathbb{G}$ with $\gamma = d/\sqrt{n}$ and $\delta = 1/(16m^2 l)$.*

### 3.4 A Randomized Programmable Hash Function

In [38] a randomized $(2,1)$-PHF was described which we now generalize to a randomzied $(m,1)$-PRF, for any $m \geq 1$.

**Definition 6.** *Let $G = (\mathbb{G}_k)$ be a group family, and $m = m(k)$ be a polynomial. In the following, let $[X]_{2^l} \in \mathbb{Z}$ denote a canonical interpretation of a field element $X \in \mathbb{F}_{2^l}$ as an integer between 0 and $2^l - 1$. We assume that $X$ and $[X]_{2^l}$ are efficiently computable from one another. Let $\mathsf{H}_{\mathsf{rand}} = (\mathsf{RPHF.Gen}, \mathsf{RPHF.Eval})$ be defined as follows.*

- $\mathsf{RPHF.Gen}(1^k)$ *returns a uniformly sampled* $\kappa = (h_0, (h_{i,j})_{(i,j)\in[2m]\times[m]}) \in \mathbb{G}^{2m^2+1}$.
- $\mathsf{RPHF.Eval}(\kappa, X; r)$ *parses* $X, r \in \mathbb{F}_{2^l}$, *and computes and returns*

$$\mathsf{RPHF.Eval}_\kappa(X;r) = h_0 \prod_{i,j=1}^{m} h_{i,j}^{([iX+r]_{2^l})^j}.$$

**Theorem 4.** *For any group $\mathbb{G}$ of known order, $\mathsf{H}_{\mathsf{rand}}$ is evasively $(m, 1, 0, 1/2)$-programmable. For the group $\mathbb{G} = \mathsf{QR}_N$ of quadratic residues modulo $N = pq$ for safe distinct primes $p$ and $q$, the function $\mathsf{H}_{\mathsf{rand}}$ is evasively $(m, 1, (2m^2 + 1)/\sqrt{N}, 1/2)$-programmable.*

The proof is given in the full version of this paper [34].

### 3.5 A Weak Programmable Hash Function

Essentially, a *weak* programmable hash function is a programmable hash function according to Definition 1, except that the trapdoor generation algorithm receives a list $X_1, \ldots, X_m \in \{0,1\}^l$ as additional input. On the one hand this allows us to construct significantly more efficient deterministic programmable hash functions, while on the other hand our generic signatures schemes are only weakly secure when instantiated with weak programmable hash functions. Fully secure signature schemes can be obtained by applying a generic conversion from weak to full security, for instance using chameleon hashes [44] which can be constructed based on standard assumptions like discrete logarithms [44], RSA [2,21,39], or factoring [44].

**Definition 7.** *A group hash function is a* weak $(m, n, \gamma, \delta)$*-programmable hash function, if there is a (probabilistic) algorithm* $\mathsf{PHF.TrapGen}$ *and a (deterministic) algorithm* $\mathsf{PHF.TrapEval}$ *such that:*

1. $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h, X_1, \ldots, X_m)$ *takes as input group elements $g, h \in \mathbb{G}$ and $X_1, \ldots, X_m \in \{0,1\}^l$, and produces a hash function key $\kappa$ together with trapdoor information $\tau$.*

*2.-4. Like in Definition 1.*

*As before, we may omit $\gamma$ and $\delta$, if $\gamma$ is negligible and $\delta$ is noticeable.* Weak evasively programmable hash functions *are defined as in Definition 2.*

Interestingly, there is a very simple way to construct a randomized programmable hash function according to Definition 3 from any weak programmable hash function. Let us now describe our instantiation of a weak (evasively) programmable hash function. This PHF already appeared implicitly in [19,49] and [9] for $m = 1$.

**Definition 8.** *Let $G = (\mathbb{G}_k)$ be a group family, and $l = l(k)$ and $m = m(k)$ be polynomials. Let $\mathsf{H}_{\mathsf{Weak}} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ be defined as follows.*

- $\mathsf{PHF.Gen}(1^k)$ *returns $\kappa = (h_0, \ldots, h_m)$, where $h_i \xleftarrow{\$} \mathbb{G}_k$ for $i \in \{0, \ldots, m\}$.*
- *On input $X \in \{0,1\}^l$ and $\kappa = (h_0, \ldots, h_m)$, $\mathsf{PHF.Eval}(\kappa, X)$ returns*

$$\mathsf{PHF.Eval}(\kappa, X) = \prod_{i=0}^{m} h_i^{(X^i)}.$$

*Here we interpret the $l$-bit strings $X_i$, $i \in [m]$, as integers in the canonical way.*

**Theorem 5.** *Let $\mathbb{G} = \mathbb{G}_k$ be a group of known order $p$. $\mathsf{H}_{\mathsf{Weak}}$ is a weak evasively $(m, 1, \gamma, \delta)$-programmable hash function with $\gamma = 0$ and $\delta = 1$.*

Again we can generalize Theorem 5 to groups of hidden order. The proof proceeds exactly like the proof of Theorem 5, except that we have to approximate the group order. For the group of quadratic residues $\mathsf{QR}_n$, we can sample the random exponents $b_i$ from $\mathbb{Z}_{n^2}$ for $i \in [0, m]$, which yields the following theorem.

**Theorem 6.** *Let $\mathbb{G} = \mathsf{QR}_N$ be the group of quadratic residues modulo $N = pq$, where $p$ and $q$ are safe distinct primes. $\mathsf{H}_{\mathsf{Weak}}$ is a $(m, 1, \gamma, \delta)$-programmable hash function over $\mathbb{G}$ with $\gamma = (m+1)/\sqrt{N}$ and $\delta = 0$.*

## 4 Signatures from the RSA Problem

### 4.1 Construction

Let $l = l(k)$ and $\lambda = \lambda(k)$ be polynomials. Let $\mathsf{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ be group hash functions over $\mathbb{G} = \mathsf{QR}_N$ with input length $l$. We define the signature scheme $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vfy})$ as follows.

$\mathsf{Gen}(1^k)$**:** The key generation algorithm picks two large safe $k/2$-bit primes $p$ and $q$, and sets $N = pq$. Then it generates a group hash function key $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$ for the group $\mathsf{QR}_N$. Finally it chooses a random key $K$ for the pseudorandom function $\mathsf{PRF} : \{0,1\}^* \to \{0,1\}^r$ and picks $c \xleftarrow{\$} \{0,1\}^r$, where $r = \lceil \log N \rceil$. These values define a function $\mathsf{F}$ as

$$\mathsf{F}(z) = \mathsf{PRF}_K(\mu || z) \oplus c,$$

where $\mu$, called the *resolving index* of $z$, denotes the smallest positive integer such that $\mathsf{PRF}_K(\mu||z) \oplus c$ is an odd prime. Here $\oplus$ denotes the bit-wise XOR

operation, and we interpret the $r$-bit string returned by $\mathsf{F}$ as an integer in the obvious way. (The definition of $\mathsf{F}$ is the same as in [40]. It is possible to replace the $\mathsf{PRF}$ with an $2k^2$-wise independent hash function [16].) The public key is $pk = (n, \kappa, K, c)$, the secret key is $sk = (pk, p, q)$.

In the following we will write $\mathsf{H}(M)$ shorthand for $\mathsf{PHF.Eval}(\kappa, M)$, and define $\mathsf{P} : \{0,1\}^\lambda \to \mathbb{N}$ as $\mathsf{P}(s) = \prod_{i=1}^{\lambda} \mathsf{F}(s_{|i})$, where $s_{|i}$ is the $i$-th prefix of $s$, i.e., the bit string consisting of the first $i$ bits of $s$. We also define $s_{|0} = \emptyset$, where $\emptyset$ is the empty string, for technical reasons.

$\mathsf{Sign}(sk, M)$**:** On input of secret key $sk$ and message $M \in \{0,1\}^l$, the signing algorithm picks $s \xleftarrow{\$} \{0,1\}^\lambda$ uniformly random and computes

$$\sigma = \mathsf{H}(M)^{1/\mathsf{P}(s)} \bmod N,$$

where the inverse of $\mathsf{P}(s)$ is computed modulo the order $\phi(n) = (p-1)(q-1)$ of the multiplicative group $\mathbb{Z}_N^*$. The signature is $(\sigma, s) \in \mathbb{Z}_N \times \{0,1\}^\lambda$.

$\mathsf{Vfy}(pk, M, (\sigma, s))$**:** On input of $pk$, message $M$, and signature $(\sigma, s)$, return `accept` if

$$\mathsf{H}(M) = \sigma^{\mathsf{P}(s)} \bmod N.$$

Otherwise return `reject`.

**Correctness.** If $\sigma = \mathsf{H}(M)^{1/\mathsf{P}(s)}$, then we have $\sigma^{\mathsf{P}(s)} = \mathsf{H}(M)^{\mathsf{P}(s)/\mathsf{P}(s)} = \mathsf{H}(M)$.

**Theorem 7.** *Let* $\mathsf{PRF}$ *be a* $(\epsilon'', t'')$*-secure pseudo-random function and* $\mathsf{H}$ *be a* $(m, 1, \gamma, \delta)$*-evasively programmable hash function. Suppose there exists a* $(t, q, \epsilon)$*-forger* $\mathcal{F}$ *breaking the existential forgery under adaptive chosen message attacks of* $\mathsf{Sig}_{\mathsf{RSA}}[\mathsf{H}]$*. Then there exists an adversary that* $(t', \epsilon')$*-breaks the RSA assumption with* $t' \approx t$ *and* $\epsilon$ *is bounded by*

$$(q+1)\lambda\left(\frac{4r^2}{\delta}\left(\epsilon' + \frac{r}{l \cdot 2^{r-l-1}}\right) + 3\epsilon'' + \frac{r(q+1)^2\lambda^2 + 2r + 1}{2^r} + \gamma + \frac{1}{2^{r-l}}\right) + \frac{q^{m+1}}{2^{m\lambda}}$$

We only give a brief proof outline here, and refer to the full version [34] for details. As customary in proofs for similar signature schemes (e.g., [23,28,36]), we distinguish between *Type I* and *Type II* forgers. A Type I forger forges a signature of the form $(M^*, \sigma^*, s^*)$ with $s^* = s_i$ for some $i \in [q]$. (That is, a Type I forger reuses some $s_i$ from a signature query.) A *Type II* forger returns a signature with a fresh $s^*$.

It will be easiest to first describe how to treat a Type II forger $\mathcal{F}$. Recall that we need to put up a simulation that is able to generate $q$ signatures $(M_i, \sigma_i, s_i)_{i \in [q]}$ for adversarially chosen messages $M_i$. To do this, we choose all

$s_i$ in advance. We then prepare the PHF $\mathsf{H}$ using $\mathsf{PHF.TrapGen}$, but relative to generators $g$ and $h$ for which we know $\mathsf{P}(s_i)$-th roots. (That is, we set $g := \hat{g}^E$ and $h = \hat{h}^E$ for $E := \prod_i \mathsf{P}(s_i)$.) This allows to generate signatures for $\mathcal{F}$; also, by the security of the PHF $\mathsf{H}$, this change goes unnoticed by $\mathcal{F}$. However, each time $\mathcal{F}$ outputs a new signature, it essentially outputs a fresh root $g^{1/\mathsf{P}(s^*)}$ of $g$, from which we can derive a $\mathsf{P}(s^*)$-th root of $\hat{g}$. To construct an RSA adversary from this experiment, we have to embed an auxiliary given exponent $e$ into the definition of $\mathsf{P}$, such that $\hat{g}^{1/\mathsf{P}(s^*)}$ allows to derive $\hat{g}^{1/e}$. This can be done along the lines of the proof of the Hohenberger-Waters scheme [40]. Concretely, for *initially given* values $s_i$ and $e$, we can set up $\mathsf{P}$ such that (a) $e$ does not divide any $\mathsf{P}(s_i)$, but (b) for any other fixed $s^*$, the probability that $e$ divides $\mathsf{P}(s^*)$ is significant. Note that in our scheme, the $s_i$ are chosen by the signer, and thus our simulation can select them in advance. In contrast to that, the HW scheme uses the signed messages $M_i$ as arguments to $\mathsf{P}$, and thus their argument achieves only a weaker form of security in which the forger has to commit to all signature queries beforehand.

Now the proof for Type I forgers proceeds similarly, but with the additional complication that we have to prepare one or more signatures of the form $\mathsf{H}(M_i)^{1/\mathsf{P}(s_i)}$ for the same $s_i = s^*$ that $\mathcal{F}$ eventually uses in his forgery. We resolve this complication by relying on the PHF properties of $\mathsf{H}$. Namely, we first choose all $s_i$ and guess $i$ (i.e., the index of the $s_i$ with $s_i = s^*$). We then prepare $\mathsf{H}$ with generators $g, h$ such that we know all $\mathsf{P}(s_j)$th roots of $h$ (for all $j$), and all $\mathsf{P}(s_j)$th roots of $g$ *for all* $s_j \neq s_i$. Our hope is that whenever $\mathcal{F}$ asks for the signature of some $M_j$ with $s_j = s_i$, we have $\mathsf{H}(M_i) \in \langle h \rangle$, so we can compute $\mathsf{H}(M_j)^{1/\mathsf{P}(s_j)}$. At the same time, we hope that $\mathsf{H}(M^*) \notin \langle h \rangle$ has a nontrivial $g$-factor, so we can build an RSA adversary as for Type II forgers. The PHF property of $\mathsf{H}$ guarantees a significant probability that this works out, provided that there are no more than $m$ indices $j$ with $s_j = s_i$ (i.e., provided that there are no $(m+1)$-collisions). However, using a birthday bound, we can reasonably upper bound the probability of $(m+1)$-collisions.

In the full version [34] we also give a variant of our scheme which is slightly more efficient but only offers weak security. A weakly secure signature scheme can be updated to a fully secure one by using a (randomized) Chameleon Hash Function.

**Efficiency.** Given $\mathsf{P}(s)$ and $\phi(N)$, computing $\sigma = \mathsf{H}(M)^{1/\mathsf{P}(s)}$ can also be carried out by one single exponentiation. Since one single evaluation of $\mathsf{P}(\cdot)$ has to perform (expected) $\lambda r$ many primality tests (for $r$-bit primes), the dominiant part of signing and verification is to compute $\mathsf{P}(s)$, for $s \in \{0,1\}^\lambda$. Theorem 7 tells us that if $\mathsf{H}$ is a $(m,1)$-PHF we can set $\lambda = \log q + k/m$, see the full version [34] for more details.

Hohenberger and Waters [40] proposed several ways to improve the efficiency of their RSA-based signature scheme. These improvements apply to our RSA-based schemes as well. We refer to the full version [34] for details.

## References


1. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
2. Ateniese, G., de Medeiros, B.: Identity-Based Chameleon Hash and Applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
3. Barreto, P.S.L.M., Naehrig, M.: Pairing-Friendly Elliptic Curves of Prime Order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
4. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental Cryptography: The Case of Hashing and Signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993: 1st Conference on Computer and Communications Security, pp. 62–73. ACM Press (November 1993)
6. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
7. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Journal of Cryptology 21(2), 178–199 (2008)
8. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on Randomizable Ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (2011)
9. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
10. Boneh, D., Boyen, X.: Secure Identity Based Encryption without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
11. Boneh, D., Boyen, X.: Short Signatures without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
12. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology 21(2), 149–177 (2008)
13. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
14. Boyen, X.: Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)
15. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM CCS 2005: 12th Conference on Computer and Communications Security, pp. 320–329. ACM Press (November 2005)

16. Cachin, C., Micali, S., Stadler, M.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
17. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th Annual ACM Symposium on Theory of Computing, pp. 209–218. ACM Press (May 1998)
18. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
19. Chatterjee, S., Sarkar, P.: Generalization of the Selective-ID Security Model for HIBE Protocols. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 241–256. Springer, Heidelberg (2006)
20. Chaum, D., Evertse, J.-H., van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
21. Chevallier-Mames, B., Joye, M.: A Practical and Tightly Secure Signature Scheme without Hash Function. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 339–356. Springer, Heidelberg (2006)
22. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-Secure Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
23. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: ACM CCS 1999: 6th Conference on Computer and Communications Security, pp. 46–51. ACM Press (November 1999)
24. Dodis, Y., Haitner, I., Tentes, A.: On the (in)security of rsa signatures. Cryptology ePrint Archive, Report 2011/087 (2011), `http://eprint.iacr.org/`
25. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
26. Dodis, Y., Oliveira, R., Pietrzak, K.: On the Generic Insecurity of the Full Domain Hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
27. Erdös, P., Frankel, P., Furedi, Z.: Families of finite sets in which no set is covered by the union of $r$ others. Israeli Journal of Mathematics 51, 79–89 (1985)
28. Fischlin, M.: The Cramer-Shoup Strong-RSASignature Scheme Revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
29. Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures without the Random Oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
30. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM Press (May 2008)
31. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
32. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 5th edn. Oxford University Press (1979)

33. Heng, S.-H., Kurosawa, K.: $k$-Resilient Identity-Based Encryption in the Standard Model. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 67–80. Springer, Heidelberg (2004)
34. Hofheinz, D., Jager, T., Kiltz, E.: Short signatures from weaker assumptions. Cryptology ePrint Archive, Report 2011/296 (2011), http://eprint.iacr.org/
35. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
36. Hofheinz, D., Kiltz, E.: Programmable Hash Functions and their Applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
37. Hofheinz, D., Kiltz, E.: Practical Chosen Ciphertext Secure Encryption from Factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
38. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. Journal of Cryptology, 1–44 (2011)
39. Hohenberger, S., Waters, B.: Realizing Hash-and-Sign Signatures under Standard Assumptions. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 333–350. Springer, Heidelberg (2009)
40. Hohenberger, S., Waters, B.: Short and Stateless Signatures from the RSA Assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
41. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
42. Kiltz, E., Galindo, D.: Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. Theor. Comput. Sci. 410(47-49), 5093–5111 (2009)
43. Kiltz, E., Pietrzak, K., Cash, D., Jain, A., Venturi, D.: Efficient Authentication from Hard Learning Problems. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 7–26. Springer, Heidelberg (2011)
44. Krawczyk, H., Rabin, T.: Chameleon signatures. In: ISOC Network and Distributed System Security Symposium – NDSS 2000. The Internet Society (February 2000)
45. Kumar, R., Rajagopalan, S., Sahai, A.: Coding Constructions for Blacklisting Problems without Computational Assumptions. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)
46. Lyubashevsky, V., Micciancio, D.: Asymptotically Efficient Lattice-Based Digital Signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
47. Okamoto, T.: Efficient Blind and Partially Blind Signatures without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
48. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd Annual ACM Symposium on Theory of Computing, pp. 387–394. ACM Press (May 1990)
49. Sahai, A., Waters, B.R.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
50. Schäge, S.: Tight Proofs for Signature Schemes without Random Oracles. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 189–206. Springer, Heidelberg (2011)

51. Schäge, S., Schwenk, J.: A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 129–142. Springer, Heidelberg (2010)
52. Shamir, A.: On the generation of cryptographically strong pseudorandom sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
53. Waters, B.R.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)