

Short Signatures From Weaker Assumptions

Dennis Hofheinz¹, Tibor Jäger², and Eike Kiltz²

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 most efficient of these schemes rely on specific “strong” number theoretic hardness assumptions which we call Strong q -assumptions.³ 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 $\text{Sig}_{\text{RSA}}[\text{H}]$ are of the form $\text{sig}(M) = (\text{H}(M)^{1/e} \bmod N, e)$, where $N = pq$ is a public RSA modulus, $\text{H}(\cdot)$ is a $(m, 1)$ -PHF, and e is a *short* prime (chosen at random during the signing process). A given HK signature (σ, e) is verified by checking if $\sigma^e = \text{H}(M) \bmod N$. 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.⁴ Over bilinear groups there exists a similar

³ 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.

⁴ 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.

construction, $\text{Sig}_{\mathbb{S}\text{-}q\text{-DH}}[\text{H}]$, 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 $\text{sig}(M) = g^{1/P(M)} \bmod N$, where $g \in \mathbb{Z}_N^*$ is a public element and $P(M) = e_1 \cdot \dots \cdot e_{|M|}$ 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 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 $\approx 2\text{kbit}$ to the public-key size [39]. The main disadvantage of HW signatures is, however, the generation and testing of the $|M|$ primes $e_1, \dots, e_{|M|}$ necessary to compute the hash function $P(M)$. 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 $\text{Sig}_{q\text{-DH}}[\text{H}]$ are (to the best of our knowledge) the first hash-and-sign schemes from this assumption, our RSA schemes $\text{Sig}_{\text{RSA}}[\text{H}]$ and $\text{Sig}_{\text{RSA}}^*[\text{H}]$ 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 known standard-model digital signatures in bilinear groups. Remarkably, $\text{Sig}_{\text{RSA}}[\text{H}_{\text{cfs}}]$ which instantiates the Strong RSA signatures from [38] with our new $(m, 1)$ -PHF H_{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.⁵ 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 significantly. More precisely, it is possible to transform each signature scheme requiring λ primality tests into a scheme which requires only λ/c primality tests, at the cost of losing a factor of 2^{-c} in the security reduction. For example, $\text{Sig}_{\text{RSA}}^*[\text{H}_{\text{Weak}}]$ [§] with $m = 11$ and $c = 40$ is a RSA-based signature scheme which requires only

⁵ Since the complexity of finding a random μ -bit prime with error 2^{-k} is $O(k\mu^4)$, we expect that for $\mu \approx 60$ (or, equivalently, using H_{cfs} with $m \geq 6$) a full exponentiation modulo a 1024-bit integer become roughly as expensive as generating a random μ -bit prime.

a *single* primality test for signing and verification, at the cost of losing a factor of 2^{-40} in the security reduction.

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

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 $k \times \text{P}_\mu$ counts the number of random μ -bit primes that need to be generated to evaluate function $\text{P}(\cdot)$. (For $\mu \gg 60$, $1 \times \text{P}_\mu$ takes more time than $1 \times \text{Exp}$.) *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} .

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 H_{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 H_{rand} with public parameters of $O(m^2)$ group elements and small randomness space. Finally, we construct a weakly secure deterministic $(m, 1)$ -PHF H_{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 $\text{Sig}_{\text{RSA}}[\text{H}]$ are of the form

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

where s is a short random bitstring, $\text{H}(\cdot)$ is a $(m, 1)$ -PHF, and $\text{P}(s) := e_1 \dots e_{|s|}$ is the product of $|s|$ primes $e_1, \dots, e_{|s|}$, where the i th prime is uniquely determined by the i th prefix $s_{|i}$ of the randomness s . (If the PHF H is probabilistic, sig additionally contains a small random bitstring r .) Our security proof is along the lines of [38], but using P enables a reduction to the RSA assumption (Theorem 7) in the standard model. The main conceptual novelty is that we apply 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 $P(s)$ only has to generate $|s|$ distinct primes $e_1, \dots, e_{|s|}$ (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 H grow with m .

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

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

where again s is a short random bitstring, H is a $(m, 1)$ programmable hash function, and $d(\cdot)$ is a special (secret) function mapping bitstrings to \mathbb{Z}_p . Since $D(s) := g^{d(s)}$ 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 $(H(M)^{1/s} \bmod N, s)$ for a prime s . The use of a *programmable* hash function H enables very efficient schemes, whose security however cannot be reduced to the standard (non-strong) RSA problem, since a forged signature $(H(M)^{1/s^*}, s^*)$ 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 $g^{1/P(M)} \bmod N$. The special structure of 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 P is applied to messages (i.e., 160-bit strings), evaluation of P requires a large number of primality tests. We combine the best of both worlds with signatures of the form $(H(M)^{1/P(s)} \bmod N, s)$ 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 $\{s_1, \dots, s_q\}$ for the forgery, where the s_i 's are the random bitstrings used for the simulated signatures; type II forgers use a new value $s^* \notin \{s_1, \dots, s_q\}$ 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 $P(\cdot)$ such that the product $P(s^*)$ contains e .⁶ Similar to the proof of HK signatures [38], the reduction for Type I forgers makes use of the $(m, 1)$ programmability of $H(\cdot)$.

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

⁶ More precisely, when simulating a type II forger, the values s_1, \dots, s_q are known in advance to the simulator. Since $s^* \notin \{s_1, \dots, s_q\}$ there is some prefix $s_{|i}^*$ of s^* that is different from all prefixes of s_1, \dots, 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 $P(\cdot)$ such that the product $P(s^*)$ contains e .

2 Preliminaries

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

2.1 Digital signatures

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

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 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 σ_i under sk for this message.
3. The forger outputs a message m^* and signature σ^* .

\mathcal{F} wins the game, if $\text{accept} \leftarrow \text{Vfy}(pk, m^*, \sigma^*)$, that is, σ^* is a valid signature for m^* , and $m^* \neq m_i$ for all i . We say that \mathcal{F} (t, q, ϵ) -breaks the EUF-CMA security of Sig , if \mathcal{F} runs in time t , makes at most q signing queries, and has success probability ϵ . We say that Sig is EUF-CMA secure, or Sig is *fully* secure, if ϵ 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, \dots, 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, ϵ) -breaks the RSA assumption, if \mathcal{A} runs in time t and

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

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

We denote with 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 $z^e \equiv y^f \pmod{N}$, computes $x \in \mathbb{Z}_N$ satisfying $x^e \equiv y \pmod{N}$.*

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, \dots, 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, \dots, i_{m+1} such that $X_{i_1} = \dots = X_{i_{m+1}}$ 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 PHF.Gen and PHF.Eval. The probabilistic algorithm $\kappa \xleftarrow{\$} \text{PHF.Gen}(1^k)$ generates a hash key κ for security parameter k . Algorithm PHF.Eval is a deterministic algorithm, taking as input a hash function key κ and $X \in \{0, 1\}^l$, and returning $\text{PHF.Eval}(\kappa, X) \in \mathbb{G}$.

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

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

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

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

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

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

We also say that H is (m, n) -programmable for short, if γ is negligible and δ is noticeable. If H is $(1, q)$ -programmable for every polynomial $q = q(k)$, then we say that H is $(1, \text{poly})$ -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 $H = (\text{RPHF.Gen}, \text{RPHF.Eval})$ is *evasively* (m, n, γ, δ) -programmable, if it is (m, n, γ, δ) -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 κ generated by $\kappa \xleftarrow{\$} \text{PHF.TrapGen}(1^k, g, h)$, and all $X_1, \dots, X_m \in \{0, 1\}^l$ and $Z_1, \dots, Z_n \in \{0, 1\}^l$ such that $X_i \neq Z_j$ for all i, j , we have

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

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

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 RPHF.Gen and RPHF.Eval. Algorithm RPHF.Gen is probabilistic, and generates a hash key $\kappa \xleftarrow{\$} \text{RPHF.Gen}(1^k)$ for security parameter k . The deterministic algorithm RPHF.Eval takes randomness $r \in \mathcal{R}_k$ and $X \in \{0, 1\}^l$ as input, and returns a group element $\text{RPHF.Eval}(\kappa, X) \in \mathbb{G}$.

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

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

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

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

$$\Pr[a_{X_1}(r_{X_1}) = \dots = a_{X_m}(r_{X_m}) = 0 \text{ and } a_{Z_1}(\tilde{r}_1), \dots, a_{Z_n}(\tilde{r}_n) \neq 0] \geq \delta,$$

where the a_{X_i} and a_{Z_j} are the output of the trapdoor evaluation $(a_{X_i}, b_{X_i}) = \text{RPHF.TrapEval}(\tau, X_i, r_{X_i})$ and $(a_{Z_j}, b_{Z_j}) = \text{PHF.TrapEval}(\tau, Z_j, \tilde{r}_j)$, and the probability is taken over the trapdoor τ produced along with κ . Here X_i may depend on X_j and r_{X_j} for $j < i$, and the Z_1, \dots, Z_n may depend on all X_i and r_i . Again we omit γ and δ , if γ is negligible and δ 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.

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

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

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 $\text{H}_{\text{Wat}} = (\text{PHF.Gen}, \text{PHF.Eval})$ be defined as follows.

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

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

Theorem 1 (Theorem 3.6 of [36]). For any fixed polynomial $q = q(k)$ and any group with known order, H_{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 $I = \{i_1, \dots, i_m\} \subseteq [s]$, 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 $d \leq 16m^2l$ 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 $s = 2^l$, $d = 16m^2l$, and $w = d/4m$. We define a hash function $H_{\text{cfs}} = (\text{PHF.Gen}, \text{PHF.Eval})$ be as follows.

- $\text{PHF.Gen}(1^k)$ returns $\kappa = (h_1, \dots, h_d)$, 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 $\kappa = (h_1, \dots, h_d)$, $\text{PHF.Eval}(\kappa, X)$ returns

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

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

Proof. Consider the following algorithms.

- $\text{PHF.TrapGen}(1^k, g, h)$ samples d uniformly random integers $b_1, \dots, b_d \xleftarrow{\$} \mathbb{Z}_p$ 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 $i \neq t$. PHF.TrapGen returns (κ, τ) with $\tau = (t, b_1, \dots, b_d)$ and $\kappa = (h_1, \dots, h_d)$.
- On input (τ, X) , 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) .

PHF.TrapGen outputs a vector of independent and uniformly distributed group elements, thus we have $\gamma = 0$. Fix $X_1, \dots, X_m, Z \in [0, 2^l - 1]$. 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 $16m^2l$ possibilities, we have $t = t'$, and thus $a_{X_i} = 0$ and $a_Z = 1$, with probability $\delta = 1/(16m^2l)$. Finally, $a_Z = 1$ implies $\gcd(a_Z, e) = 1$ for all primes e , thus H_{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 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 $\mathbb{G} = \text{QR}_n$ be the group of quadratic residues modulo $n = pq$, where p and q are safe distinct primes. H_{cfs} is a $(m, 1, \gamma, \delta)$ -evasively programmable hash function over \mathbb{G} with $\gamma = d/\sqrt{n}$ and $\delta = 1/(16m^2l)$.

3.4 A randomized programmable hash function

In [38] a randomized $(2, 1)$ -PHF was described which we now generalize to a randomized $(m, 1)$ -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 $H_{\text{rand}} = (\text{RPHF.Gen}, \text{RPHF.Eval})$ be defined as follows.

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

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

Theorem 4. For any group \mathbb{G} of known order, H_{rand} is evasively $(m, 1, 0, 1/2)$ -programmable. For the group $\mathbb{G} = \text{QR}_N$ of quadratic residues modulo $N = pq$ for safe distinct primes p and q , the function H_{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 $X_1, \dots, X_m \in \{0, 1\}^l$ 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, γ, δ) -programmable hash function, if there is a (probabilistic) algorithm PHF.TrapGen and a (deterministic) algorithm PHF.TrapEval such that:

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

2.-4. Like in Definition 1.

As before, we may omit γ and δ , if γ is negligible and δ 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 $\text{H}_{\text{Weak}} = (\text{PHF.Gen}, \text{PHF.Eval})$ be defined as follows.

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

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

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 . H_{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 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 $\mathbb{G} = \text{QR}_N$ be the group of quadratic residues modulo $N = pq$, where p and q are safe distinct primes. H_{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 $\text{H} = (\text{PHF.Gen}, \text{PHF.Eval})$ be group hash functions over $\mathbb{G} = \text{QR}_N$ with input length l . We define the signature scheme $\text{Sig}_{\text{RSA}}[\text{H}] = (\text{Gen}, \text{Sign}, \text{Vfy})$ as follows.

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 $\kappa \xleftarrow{\$} \text{PHF.Gen}(1^k)$ for the group QR_N . Finally it chooses a random key K for the pseudorandom function $\text{PRF} : \{0, 1\}^* \rightarrow \{0, 1\}^r$ and picks $c \xleftarrow{\$} \{0, 1\}^r$, where $r = \lceil \log N \rceil$. These values define a function F as

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

where μ , called the *resolving index* of z , denotes the smallest positive integer such that $\text{PRF}_K(\mu || z) \oplus c$ is an odd prime. Here \oplus denotes the bit-wise XOR operation, and we interpret the r -bit string returned by F as an integer in the obvious way. (The definition of F is the same as in [40]. It is possible to replace the 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 $\text{H}(M)$ shorthand for $\text{PHF.Eval}(\kappa, M)$, and define $\text{P} : \{0, 1\}^\lambda \rightarrow \mathbb{N}$ as $\text{P}(s) = \prod_{i=1}^\lambda \text{F}(s|_i)$, 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.

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

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

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

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

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

Otherwise return **reject**.

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

Theorem 7. Let PRF be a (ϵ'', t'') -secure pseudo-random function and H be a $(m, 1, \gamma, \delta)$ -evasively programmable hash function. Suppose there exists a (t, q, ϵ) -forger \mathcal{F} breaking the existential forgery under adaptive chosen message attacks of $\text{Sig}_{\text{RSA}}[H]$. Then there exists an adversary that (t', ϵ') -breaks the RSA assumption with $t' \approx t$ and ϵ 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^*, σ^*, 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 H using PHF.TrapGen , but relative to generators g and h for which we know $P(s_i)$ -th roots. (That is, we set $g := \hat{g}^E$ and $h = \hat{h}^E$ for $E := \prod_i P(s_i)$.) This allows to generate signatures for \mathcal{F} ; also, by the security of the PHF H , this change goes unnoticed by \mathcal{F} . However, each time \mathcal{F} outputs a new signature, it essentially outputs a fresh root $g^{1/P(s^*)}$ of g , from which we can derive a $P(s^*)$ -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 P , such that $\hat{g}^{1/P(s^*)}$ 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 P such that (a) e does not divide any $P(s_i)$, but (b) for any other fixed s^* , the probability that e divides $P(s^*)$ 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 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 $H(M_i)^{1/P(s_i)}$ 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 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 H with generators g, h such that we know all $P(s_j)$ th roots of h (for all j), and all $P(s_j)$ 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 $H(M_i) \in \langle h \rangle$, so we can compute $H(M_j)^{1/P(s_j)}$. At the same time, we hope that $H(M^*) \notin \langle h \rangle$ has a nontrivial g -factor, so we can build an RSA adversary as for Type II forgers. The PHF property of 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 $P(s)$ and $\phi(N)$, computing $\sigma = H(M)^{1/P(s)}$ can also be carried out by one single exponentiation. Since one single evaluation of $P(\cdot)$ has to perform (expected) λr many primality tests (for r -bit primes), the dominant part of signing and verification is to compute $P(s)$, for $s \in \{0, 1\}^\lambda$. Theorem 7 tells us that if 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. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, May 2010.

2. Giuseppe Ateniese and Breno de Medeiros. Identity-based chameleon hash and applications. In Ari Juels, editor, *FC 2004: 8th International Conference on Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 164–180. Springer, February 2004.
3. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, August 2005.
4. Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO’94*, volume 839 of *Lecture Notes in Computer Science*, pages 216–233. Springer, August 1994.
5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.
6. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT’96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, May 1996.
7. Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, 21(2):178–199, April 2008.
8. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422. Springer, March 2011.
9. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.
10. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, August 2004.
11. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, May 2004.
12. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
13. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, December 2001.
14. Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 499–517. Springer, May 2010.
15. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 320–329. ACM Press, November 2005.
16. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT’99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, May 1999.
17. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.
18. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, May 2010.
19. Sanjit Chatterjee and Palash Sarkar. Generalization of the selective-ID security model for HIBE protocols. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 241–256. Springer, April 2006.
20. David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology – EUROCRYPT’87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer, April 1988.
21. Benoît Chevallier-Mames and Marc Joye. A practical and tightly secure signature scheme without hash function. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 339–356. Springer, February 2007.
22. Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, December 2007.

23. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 46–51. ACM Press, November 1999.
24. Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. On the (in)security of rsa signatures. Cryptology ePrint Archive, Report 2011/087, 2011. <http://eprint.iacr.org/>.
25. Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, April / May 2002.
26. Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 449–466. Springer, August 2005.
27. P. Erdős, P. Frankel, and Z. Füredi. 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. Marc Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer, January 2003.
29. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT’99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, May 1999.
30. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.
31. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
32. G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, fifth edition, 1979.
33. Swee-Huay Heng and Kaoru Kurosawa. k -resilient identity-based encryption in the standard model. In Tatsuaiki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 67–80. Springer, February 2004.
34. Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. Cryptology ePrint Archive, Report 2011/296, 2011. <http://eprint.iacr.org/>.
35. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer, August 2007.
36. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, August 2008.
37. Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 313–332. Springer, April 2009.
38. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *Journal of Cryptology*, pages ???–???, 2011.
39. Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350. Springer, April 2009.
40. Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 654–670. Springer, August 2009.
41. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, March 2006.
42. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theor. Comput. Sci.*, 410(47-49):5093–5111, 2009.
43. Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 7–26. Springer, May 2011.
44. Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *ISOC Network and Distributed System Security Symposium – NDSS 2000*. The Internet Society, February 2000.
45. Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 609–623. Springer, August 1999.
46. Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, March 2008.

47. Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99. Springer, March 2006.
48. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, May 1990.
49. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, May 2005.
50. Sven Schäge. Tight proofs for signature schemes without random oracles. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 189–206. Springer, May 2011.
51. Sven Schäge and Jörg Schwenk. A CDH-based ring signature scheme with short signatures and public keys. In Radu Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 129–142. Springer, January 2010.
52. Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, 1(1):38–44, 1983.
53. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.