Programmable Hash Functions and Their Applications

Dennis Hofheinz[⋆] and Eike Kiltz[⋆⋆]

Cryptology and Information Security Research Theme
CWI Amsterdam, The Netherlands
{hofheinz,kiltz}@cwi.nl

Abstract. We introduce a new information-theoretic primitive called *programmable hash functions* (PHFs). PHFs can be used to *program* the output of a hash function such that it contains solved or unsolved discrete logarithm instances with a certain probability. This is a technique originally used for security proofs in the random oracle model. We give a variety of *standard model* realizations of PHFs (with different parameters).

The programmability of PHFs make them a suitable tool to obtain black-box proofs of cryptographic protocols when considering adaptive attacks. We propose generic digital signature schemes from the strong RSA problem and from some hardness assumption on bilinear maps that can be instantiated with any PHF. Our schemes offer various improvements over known constructions. In particular, for a reasonable choice of parameters, we obtain short standard model digital signatures over bilinear maps.

1 Introduction

1.1 Programmable Hash Functions

A group hash function is an efficiently computable function that maps binary strings into a group $\mathbb{G}$. We propose the concept of a *programmable hash function* which is a keyed group hash function that can behave in two indistinguishable ways, depending on how the key is generated. If the standard key generation algorithm is used, then the hash function fulfills its normal functionality, i.e., it properly hashes its inputs into a group $\mathbb{G}$. The alternative (trapdoor) key generation algorithm outputs a key that is *indistinguishable* from the one output by the standard algorithm. It furthermore generates some additional secret trapdoor information that depends on two generators g and h from the group. This trapdoor information makes it possible to relate the output of the hash function to g and h: for any input X, one obtains integers a_X and b_X such that the relation $\mathrm{H}(X) = g^{a_X} h^{b_X} \in \mathbb{G}$ holds. For the PHF to be (m, n)-programmable

⋆ Supported by the Dutch Organization for Scientific Research (NWO).

⋆⋆ Supported by the research program Sentinels (http://www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

we require that *for all* choices of $X_1, \ldots, X_m$ and $Z_1, \ldots, Z_n$ with $X_i \neq Z_j$, it holds that $a_{X_i} = 0$ but $a_{Z_j} \neq 0$, with some non-negligible probability. Hence parameter m controls the number of elements X for which we can hope to have $\mathrm{H}(X) = h^{b_X}$; parameter n controls the number of elements Z for which we can hope to have $\mathrm{H}(Z) = g^{a_Z} h^{b_Z}$ for some $a_Z \neq 0$.

The concept becomes useful in groups with hard discrete logarithms and when the trapdoor key generation algorithm does not know the discrete logarithm of h to the basis g. It is then possible to program the hash function such that the hash images of all possible choices $X_1, \ldots, X_m$ of m inputs are $\mathrm{H}(X_i) = h^{b_{X_i}}$, i.e., they do not depend on g (since $a_{X_i} = 0$). At the same time the hash images of all possible choices $Z_1, \ldots, Z_n$ of n (different) inputs are $\mathrm{H}(Z_i) = g^{b_{Z_i}} \cdot h^{b_{Z_i}}$, i.e., they do depend on g in a known way (since $a_{Z_i} \neq 0$). Intuitively, this resembles a scenario we are often confronted with in "provable security": for some of the hash outputs we know the discrete logarithm, and for some we do not. This situation appears naturally during a reduction that involves an adaptive adversary. Concretely, knowledge of the discrete logarithms of some hash queries can be used to simulate, e.g., a signing oracle for an adversary (which would normally require knowledge of a secret signing key). On the other hand, once the adversary produces, e.g., a signature on its own, our hope is that this signature corresponds to a hash query for which the we do *not* know the discrete logarithm. This way, the adversary has produced a piece of nontrivial secret information which can be used to break an underlying computational assumption.

This way of "programming" a hash function is very popular in the context of random oracles [3] (which, in a sense, are ideally programmable hash functions), and has been used to derive proofs of the adaptive security of simple signature schemes [4]. An (m, poly)-PHF is a (m, n)-PHF for all polynomials n. A (poly, m)-PHF is defined the same way. Using this notation, a random oracle implies a $(\mathsf{poly}, 1)$-PHF.[1]

Instantiations. As our central instantiation of a PHF we use the following function which was originally introduced by Chaum et. al. [13] as a collision-resistant hash function. The "multi-generator" hash function $\mathrm{H}^{\mathsf{MG}} : \{0,1\}^\ell \to \mathbb{G}$ is defined as $\mathrm{H}^{\mathsf{MG}}(X) := h_0 \prod_{i=1}^{\ell} h_i^{X_i}$, where the h_i are public generators of the group and $X = (X_1, \ldots, X_\ell)$. After its discovery in [13] it was also used in other constructions (e.g., [11,14,2,25]), relying on other useful properties beyond collision resistance. Specifically, in the analysis of his identity-based encryption scheme, Waters [25] implicitly proved that, using our notation, H^{MG} is a $(1, \mathsf{poly})$-programmable hash function.

Our main result concerning instantiations of PHFs is a new analysis of H^{MG} showing that it is also a $(2, 1)$-PHF. Furthermore, we can use our new techniques to prove better bounds on the $(1, \mathsf{poly})$-programmability of H^{MG}. We stress that

[1] By "programming" the random oracle as $\mathrm{H}(X) = g^{a_X} h^{b_X}$ (for random a_X, b_X) with some sufficiently small but noticeable probability p and $\mathrm{H}(X) = h^{b_X}$ with probability $1 - p$ [16].

our analysis uses random walk techniques and is different from the one implicitly given in [25].

Unfortunately, the PHF H^{MG} has a relatively large public evaluation key. (Its key consists of $\ell + 1$ group elements.) In our main application, signature schemes, this will lead to a tradeoff between public key size and signature size: using PHFs decreases the signature size, at the price of an increased public key size. See below for more details.

Variations. The concept of PHFs can be extended to randomized programmable hash functions (RPHFs). An RPHF is like a PHF whose input takes an additional parameter, the randomness. Our main construction of a randomized hash function is $\mathrm{RH}^{\mathsf{Poly}_m}$, which is $(m, 1)$-programmable. Note that unlike H^{MG}, the construction of the hash function depends on the parameter m. In particular, the complexity of $\mathrm{RH}^{\mathsf{Poly}_m}$ grows quadratically in m.

In some applications (e.g., for RSA signatures) we need a special type a PHF which we call bounded PHF. Essentially, for bounded PHFs we need to know a certain upper bound on the $|a_X|$, for all X. Whereas H^{MG} is bounded, $\mathrm{RH}^{\mathsf{Poly}_m}$ is only bounded for $m = 1$.

1.2 Applications

Collision Resistant Hashing. We aim to use PHFs as a tool to provide black-box proofs for various cryptographic protocols. As a toy example let us sketch why, in prime-order groups with hard discrete logarithms, any $(1, 1)$-PHF implies collision resistant hashing. Setting up H using the trapdoor generation algorithm will remain unnoticed for an adversary, but any collision $\mathrm{H}(X) = \mathrm{H}(Z)$ with $X \neq Z$ gives rise to an equation $g^{a_X} h^{b_X} = \mathrm{H}(X) = \mathrm{H}(Z) = g^{a_Z} h^{b_Z}$ with known exponents. Since the hash function is $(1, 1)$-programmable we have that, with non-negligible probability, $a_X = 0$ and $a_Z \neq 0$. This implies $g = h^{(b_X - b_Z)/a_Z}$, revealing the discrete logarithm of h to the base g.

Generic Bilinear Map signatures. We propose the following generic Bilinear Maps signature scheme with respect to a group hash function H. The signature of a message X is defined as the tuple

$$\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]: \quad sig = (\mathrm{H}(X)^{\frac{1}{x+s}}, s) \in \mathbb{G} \times \{0, 1\}^{\eta}, \tag{1}$$

where s is a random η bit-string. Here $x \in \mathbb{Z}_{|\mathbb{G}|}$ is the secret key. The signature can be verified with the help of the public key g, g^x and a bilinear map. Our main theorem concerning the Bilinear Map signatures states that if, for some $m \geq 1$, H is an $(m, 1)$-programmable hash function and the q-Strong Diffie-Hellman (q-SDH) assumption [6] holds, then the above signature scheme is unforgeable against chosen message attacks [23]. Here, the parameter m controls the size $\eta = \eta(m)$ of the randomness s. For "80-bit security" and assuming the scheme establishes no more than $q = 2^{30}$ signatures [4], we can choose $\eta = 30 + 80/m$ such that $\eta = 70$ is sufficient when using our $(2, 1)$-PHF H^{MG}. The total signature size amounts to $160 + 70 = 230$ bits. (See below for details.) Furthermore, our generic

Bilinear Map scheme can also be instantiated with any randomized PHF. Then the signature of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{RH}]$ is defined as $sig := (\mathrm{RH}(X;r)^{1/(x+s)}, s, r)$, where r is chosen from the PRHF's randomness space.

Generic RSA signatures. We propose the following generic RSA signature scheme with respect to a group hash function H. The signature of a message X is defined as the tuple

$$\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]: \quad sig = (\mathrm{H}(X)^{1/e}, r) \in \mathbb{Z}_N \times \{0,1\}^\eta, \tag{2}$$

where e is a η bit prime. The eth root can be computed using the factorization of $N = pq$ which is contained in the secret key. Our main theorem concerning RSA signatures states that if, for some $m \geq 1$, H is a bounded $(m, 1)$-programmable hash function and the strong RSA assumption holds, then the above signature scheme is unforgeable against chosen message attacks. Again, the parameter m controls the size of the prime as $\eta \approx 30 + 80/m$. Our generic RSA scheme can also be instantiated with a bounded randomized PHF.

Other applications. BLS signatures [8,9] are examples of "full-domain hash" (FDH) signature schemes [4]. Using the properties of a $(\mathsf{poly}, 1)$-programmable hash function, one can give a black-box reduction from unforgeability of $\mathsf{SIG}_{\mathrm{BLS}}$ to breaking the CDH assumption. The same reduction also holds for all full-domain hash signatures, for example also RSA-FDH [4]. Unfortunately, we do not know of any standard-model instantiation of $(\mathsf{poly}, 1)$-PHFs. This fact may be not too surprising given the impossibility results from [18].[2]

It is furthermore possible to reduce the security of Waters signatures [25] to breaking the CDH assumption, when instantiated with a $(1, \mathsf{poly})$-programmable hash function. This explains Waters' specific analysis in our PHF framework. Furthermore, our improved bound on the $(1, \mathsf{poly})$-programmability of H^{MG} gives a (slightly) tighter security reduction for Waters IBE and signature scheme.

1.3 Short Signatures

Our main application of PHFs are short signatures in the standard model. We now discuss our results in more detail. We refer to [9,6] for applications of short signatures.

The birthday paradox and randomized signatures. A signature scheme $\mathsf{SIG}_{\mathrm{Fisch}}$ by Fischlin [19] (itself a variant of the RSA-based Cramer-Shoup signatures [17]) is defined as follows. The signature for a message m is given by $sig := (e, r, (h_0 h_1^r h_2^{m+r \bmod 2^\ell})^{1/e} \bmod N)$, where e is a random η-bit prime and r is a random ℓ bit mask. The birthday paradox (for uniformly sampled primes)

[2] We remark that the impossibility results from [18] do not imply that $(\mathsf{poly}, 1)$-programmable hash functions do not exist since they only rule out the possibility of proving the security of such constructions based on any assumption which is satisfied by random functions, thus it might still be possible to construct such objects using, say homomorphic properties.

tells us that after establishing q distinct Fischlin signatures, the probability that there exist two signatures, (e, r_1, y_1) on m_1 and (e, r_2, y_2) on m_2, with the *same prime* e is roughly $q^2\eta/2^\eta$. One can verify that in case of a collision, $(e, 2r_1 - r_2, 2y_1 - y_2)$ is a valid signature on the "message" $2m_1 - m_2$ (with constant probability). Usually, for "k bit security" one requires the adversary's success ratio (i.e., the forging probability of an adversary divided by its running time) to be upper bounded by 2^{-k}. For $k = 80$ and assuming the number of signature queries is upper bounded by $q = 2^{30}$, the length of the prime must therefore be at least $\eta > 80+30 = 110$ bits to immunize against this birthday attack. We remark that for a different, more technical reason, Fischlin's signatures even require $\eta \geq 160$ bits.

Beyond the birthday paradox. In fact, Fischlin's signature scheme can be seen as our generic RSA signatures scheme from (2), instantiated with a concrete $(1,1)$-RPHF ($\mathrm{RH}^{\mathsf{Poly}_1}$). In our notation, the programmability of the hash function is used at the point where an adversary uses a given signature (e, y_1) to create a forgery (e, y) with the *same prime* e. A simulator in the security reduction has to be able to compute $y_1 = \mathrm{H}(X)^{1/e}$ but must use $y = \mathrm{H}(Z)^{1/e}$ to break the strong RSA challenge, i.e., to compute $g^{1/e'}$ and $e' > 1$ from g. However, since the hash function is $(1,1)$-programmable we can program H with g and $h = g^e$ such that, with some non-negligible probability, $\mathrm{H}(X)^{1/e} = h^{b_X} = g^{b_{X_1}}$ can be computed but $\mathrm{H}(Z)^{1/e} = (g^{a_Z} h^{b_Z})^{1/e} = g^{a_Z/e} g^{b_Z}$ can be used to break the strong RSA assumption since $a_Z \neq 0$.

Our central improvement consists of instantiating the generic RSA signature scheme with a $(m,1)$-PHF to break the birthday bound. The observation is that such hash functions can guarantee that after establishing up to m signatures with respect to the same prime, forging is still impossible. In analogy to the above, with a $(m,1)$-PHF the simulation is successful as long as there are at most m many signatures that use the same prime as in the forgery. By the generalized birthday paradox we know that after establishing q distinct generic RSA signatures the probability that there exists m signatures with the same prime is roughly $q^{m+1}(\frac{\eta}{2^\eta})^m$. Again, the success ration has to be bounded by 2^{-80} for $q = 2^{30}$ which means that $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$ instantiated with a $(2,1)$-PHF can have primes as small as $\eta = 80$ bits to be provably secure.

The security proof for the bilinear map scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ is similar. Due to the extended birthday paradox (for uniform random strings), $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ instantiated with a $(m,1)$-PHF only needs $\eta = 30 + 80/m$ bits of randomness to be provably secure. For example, with our $(2,1)$-PHF H^{MG} we need 70 bits of randomness.

Comparison. Table 1 compares the signature sizes of our and known signatures assuming $q = 2^{30}$. For RSA signatures our scheme $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}^{\mathsf{MG}}]$ offers a short alternative to Fischlin's signature scheme. More importantly, generating a random 80 bit prime will be considerably faster than a 160 bit one. We expect that, compared to the one by Fischlin, our new scheme roughly halves the signing time.

The main advantage of our bilinear maps scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$ is its very compact signatures of only 230 bits. This saves 90 bits compared to the short

Scheme	Type	Signature Size	Key Size
Boneh-Boyen [6]	Bilinear	$\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert = 320$	$2\lvert\mathbb{G}\rvert = 320$
Ours: $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$	Bilinear	$\lvert\mathbb{G}\rvert + \lvert s\rvert = 230$	$(\ell+2)\lvert\mathbb{G}\rvert = 26\mathrm{k}$
Cramer-Shoup [17]	RSA	$2 \cdot \lvert\mathbb{Z}_N\rvert + \lvert e\rvert = 2208$	$3 \cdot \lvert\mathbb{Z}_N\rvert + \lvert e\rvert = 3232$
Fischlin [19] (=$\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{RH}^{\mathsf{Poly}_1}]$)	RSA	$\lvert\mathbb{Z}_N\rvert + \lvert r\rvert + \lvert e\rvert = 1344$	$4 \cdot \lvert\mathbb{Z}_N\rvert = 4096$
Ours: $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}^{\mathsf{MG}}]$	RSA	$\lvert\mathbb{Z}_N\rvert + \lvert e\rvert = 1104$	$(\ell+1)\lvert\mathbb{Z}_N\rvert = 164\mathrm{k}$

Table 1. Recommended signature sizes of different schemes. The parameters are chosen to provide unforgeability with $k = 80$ bits security after revealing maximal $q = 2^{30}$ signatures. RSA signatures are instantiated with a modulus of $|N| = 1024$ bits, bilinear maps signatures in asymmetric pairings with $|\mathbb{G}| = \log p = 160$ bits. We assume without loss of generality that messages are of size ℓ bits (otherwise, we can apply a collision-resistant hash function first), where ℓ must be in the order of $2k = 160$ in order to provide k bits of security.

signatures scheme from Boneh-Boyen [6,7] and is only 70 bits larger than the random oracle BLS signatures. However, a drawback of our constructions is the size of the verification key since it includes the group hash key κ. For example, for $\mathrm{H}^{\mathsf{MG}} : \{0,1\}^\ell \to \mathbb{G}$, κ contains $\ell + 1$ group elements, where $\ell = 160$. Concretely, that makes a verification key of $26k$ bits compared to 320 bits from [6].

We remark that our concrete security reductions for the two generic schemes are not tight, i.e., the reductions roughly lose $\log(q/\delta)$ bits of security (cf. Theorems 10 and 13). Strictly speaking, a non-tight reduction has to be penalized by having to choose a larger group order. Even though this is usually not done in the literature [17,19], we also consider concrete signature size when additionally taking the non-tight security reduction into account. Since all known RSA schemes [17,19] have the same non-tight reduction as we have, we only consider schemes based on bilinear maps. A rigorous comparison appears in the full version.

Related Signature Schemes. Our generic bilinear map signature scheme belongs to the class of "inversion-based" signature schemes originally proposed in [24] and first formally analyzed in [6]. Other related standard-model schemes can be found in [22,10]. We stress that our signatures derive from the above since the message does not appear in the denominator of the exponent. This is an essential feature to get around the birthday bounds. Our generic RSA signature scheme builds on [19] which itself is based on the early work by Cramer and Shoup [17]. Other standard-model RSA schemes are [21,12,26,15].

1.4 Open problems

We show that PHFs provide a useful primitive to obtain black-box proofs for certain signature schemes. We leave it for future research to extend the application of PHFs to other types of protocols.

We leave it as an open problem to prove or disprove the standard-model existence of $(\mathsf{poly}, 1)$-RPHFs. (Note that a positive result would imply a security proof for FDH signatures like [9]). Moreover, we are asking for a concrete construction of a deterministic $(3, 1)$-PHF that would make it possible to shrink the signature size of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ to ≈ 215 bits. A *bounded* $(10, 1)$-RPHF would make it possible to shrink the size of the prime in $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{RH}]$ to roughly 40 bits. This is interesting since generating random 40 bit primes is very inexpensive. Finally, a $(2, 1)$ or $(1, \mathsf{poly})$-PHF with more compact parameters would have dramatic impact on the practicability of our signature schemes or Waters' IBE scheme [25].

2 Preliminaries

Notation. If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. For $n \in \mathbb{N}$, we write $[n]$ shorthand for $\{1, \ldots, n\}$. If S is a set then $s \stackrel{\$}{\leftarrow} S$ denotes the operation of picking an element s of S uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \stackrel{\$}{\leftarrow} \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output. With PPT we denote probabilistic polynomial time. For random variables X and Y, we write $X \stackrel{\gamma}{\equiv} Y$ if their statistical distance is at most γ.

Digital Signatures. A digital signature scheme SIG consists of the PPT algorithms. The key generation algorithm generates a secret signing and a public verification key. The signing algorithm inputs the signing key and a message and returns a signature. The deterministic verification algorithm inputs the verification key and returns accept or reject. We demand the usual correctness properties. We recall the definition for unforgeability against chosen-message attacks (UF-CMA), played between a challenger and a forger $\mathcal{F}$:

1. The challenger generates verification/signing key, and gives the verification key to $\mathcal{F}$;
2. $\mathcal{F}$ makes a number of *signing queries* to the challenger; each such query is a message m_i; the challenger signs m_i, and sends the result sig_i to $\mathcal{F}$;
3. $\mathcal{F}$ outputs a message m and a signature sig.

We say that forger $\mathcal{F}$ wins the game if sig is a valid signature on m and it has not queried a signature on m before. Forger $\mathcal{F}$ (t, q, ϵ)-breaks the UF-CMA security of SIG if its running time is bounded by t, it makes at most Q signing queries, and the probability that it wins the above game is bounded by ϵ. Finally, SIG is UF-CMA secure if no forger can (t, q, ϵ)-break the UF-CMA security of SIG for polynomial t and q and non-negligible ϵ.

Pairing groups and the q-SDH assumption. Our pairing schemes will be defined on families of bilinear groups $(\mathbb{PG}_k)_{k \in \mathbb{N}}$. A pairing group $\mathbb{PG} = \mathbb{PG}_k = (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g)$ consist of a multiplicative cyclic group $\mathbb{G}$ of prime order p, where

$2^k < p < 2^{k+1}$, a multiplicative cyclic group $\mathbb{G}_T$ of the same order, a generator $g \in \mathbb{G}$, and a non-degenerate bilinear pairing $\hat{e}$: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. See [6] for a description of the properties of such pairings. We say an adversary $\mathcal{A}$ (t, ϵ)-breaks the q-strong Diffie-Hellman (q-SDH) assumption if its running time is bounded by t and

$$\Pr[(s, g^{\frac{1}{x+s}}) \stackrel{\$}{\leftarrow} \mathcal{A}(g, g^x, \ldots, g^{x^q})] \geq \epsilon,$$

where $g \stackrel{\$}{\leftarrow} \mathbb{G}_T$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. We require that in $\mathbb{PG}$ the q-SDH [5] assumption holds meaning that no adversary can (t, ϵ) break the q-SDH problem for a polynomial t and non-negligible ϵ.

RSA groups and the strong RSA assumption. Our RSA schemes will be defined on families of RSA groups $(\mathbb{RG}_k)_{k \in \mathbb{N}}$. A safe RSA group $\mathbb{RG} = \mathbb{RG}_k = (p, q)$ consists of two distinct safe prime p and q of $k/2$ bits. Let QR_N denote the cyclic group of quadratic residues modulo an RSA number $N = pq$. We say an adversary $\mathcal{A}$ (t, ϵ)-breaks the strong RSA assumption if its running time is bounded by t and

$$\Pr[(e > 1, z^{1/e}) \stackrel{\$}{\leftarrow} \mathcal{A}(N = pq, z)] \geq \epsilon,$$

where $z \stackrel{\$}{\leftarrow} \mathbb{Z}_N$. We require that in $\mathbb{RG}$ the strong RSA assumption [1,20] holds meaning that no adversary can (t, ϵ)-break the strong RSA problem for a polynomial t and non-negligible ϵ.

3 Programmable Hash Functions

A *group family* $G = (\mathbb{G}_k)$ is a family of cyclic groups $\mathbb{G}_k$, indexed by the security parameter $k \in \mathbb{N}$. When the reference to the security parameter k is clear, we will simply write $\mathbb{G}$ instead of $\mathbb{G}_k$. A *group hash function* $\mathrm{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ for a group family $G = (\mathbb{G}_k)$ and with input length $\ell = \ell(k)$ consists of two PPT algorithms. For security parameter $k \in \mathbb{N}$, a key $\kappa \stackrel{\$}{\leftarrow} \mathsf{PHF.Gen}(1^k)$ is generated by the key generation algorithm $\mathsf{PHF.Gen}$. This key κ can then be used for the deterministic evaluation algorithm $\mathsf{PHF.Eval}$ to evaluate H via $y \leftarrow \mathsf{PHF.Eval}(\kappa, X) \in \mathbb{G}$ for any $X \in \{0,1\}^\ell$. We write $\mathrm{H}_\kappa(X) = \mathsf{PHF.Eval}(\kappa, X)$.

Definition 1. *A group hash function* H *is an* (m, n, γ, δ)-programmable hash function *if there are PPT algorithms* $\mathsf{PHF.TrapGen}$ *(the trapdoor key generation algorithm) and* $\mathsf{PHF.TrapEval}$ *(the deterministic trapdoor evaluation algorithm) such that the following holds:*

Syntactics: *For group elements $g, h \in \mathbb{G}$, the trapdoor key generation $(\kappa', t) \stackrel{\$}{\leftarrow} \mathsf{PHF.TrapGen}(1^k, g, h)$ produces a key κ' along with a trapdoor t. Moreover, $(a_X, b_X) \leftarrow \mathsf{PHF.TrapEval}(t, X)$ produces $a_X, b_X \in \mathbb{Z}$ for any $X \in \{0,1\}^\ell$.*

Correctness: *We demand $\mathrm{H}_{\kappa'}(X) = \mathsf{PHF.Eval}(\kappa', X) = g^{a_X} h^{b_X}$ for all generators $g, h \in \mathbb{G}$ and all possible $(\kappa', t) \stackrel{\$}{\leftarrow} \mathsf{PHF.TrapGen}(1^k, g, h)$, for all $X \in \{0,1\}^\ell$ and the corresponding $(a_X, b_X) \leftarrow \mathsf{PHF.TrapEval}(t, X)$.*

Statistically close trapdoor keys: *For all generators $g, h \in \mathbb{G}$ and for $\kappa \stackrel{\$}{\leftarrow}$ $\mathsf{PHF.Eval}(1^k)$ and $(\kappa', t) \stackrel{\$}{\leftarrow} \mathsf{PHF.Eval}(1^k, g, h)$, the keys κ and κ' are statistically γ-close: $\kappa \stackrel{\gamma}{\equiv} \kappa'$.*

Well-distributed logarithms: *For all generators $g, h \in \mathbb{G}$ and all possible κ' in the range of (the first output component of) $\mathsf{PHF.TrapGen}(1^k, g, h)$, for all $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^\ell$ such that $X_i \neq Z_j$ for any i, j, and for the corresponding $(a_{X_i}, b_{X_i}) \leftarrow \mathsf{PHF.TrapEval}(t, X_i)$ and $(a_{Z_i}, b_{Z_i}) \leftarrow \mathsf{PHF.TrapEval}(t, Z_i)$, we have*

$$\mathsf{Pr}\left[a_{X_1} = \ldots = a_{X_m} = 0 \quad \wedge \quad a_{Z_1}, \ldots, a_{Z_n} \neq 0\right] \geq \delta, \tag{3}$$

where the probability is over the trapdoor t that was produced along with κ'.

We simply say that H *is an* (m, n)-programmable hash function *if there is a negligible γ and a noticeable δ such that* H *is (m, n, γ, δ)-programmable. Furthermore, we call* H (poly, n)-programmable *if* H *is (q, n)-programmable for every polynomial $q = q(k)$. We say that* H *is* (m, poly)-programmable *(resp.* $(\mathsf{poly}, \mathsf{poly})$-programmable*) if the obvious holds.*

Note that a group hash function can be a (m, n)-programmable hash function for different parameters m, n with different trapdoor key generation and trapdoor evaluation algorithms.

In our RSA application, the following additional definition will prove useful:

Definition 2. *In the situation of Definition 1, we say that* H *is β-bounded (m, n, γ, δ)-programmable if $|a_x| \leq \beta(k)$ always.*

As a first example, note that a (programmable) random oracle $\mathcal{O}$ (i.e., a random oracle which we can completely control during a proof) is trivially a $(1, \mathsf{poly})$ or $(\mathsf{poly}, 2)$-programmable hash function: given generators g and h, we simply define the values $\mathcal{O}(X_i)$ and $\mathcal{O}(Z_j)$ in dependence of the X_i and Z_j as suitable expressions $g^a h^b$. (For example, by using Coron's method [16].)

As already mentioned in the introduction, we can show a positive and natural result with a similar reduction on the discrete logarithm problem: any (non-trivially) programmable hash function is collision-resistant (a proof appears in the full version).

Theorem 3. *Assume $|\mathbb{G}|$ is known and prime, and the discrete logarithm problem in $\mathbb{G}$ is hard. Let* H *be a $(1, 1)$-programmable hash function. Then* H *is collision-resistant.*

We will now give an example of a programmable hash function in the standard model.

Definition 4. *Let $G = (\mathbb{G}_k)$ be a group family, and let $\ell = \ell(k)$ be a polynomial. Then,* $\mathrm{H}^{\mathsf{MG}} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ *is the following group hash function:*

– $\mathsf{PHF.Gen}(1^k)$ *returns a uniformly sampled $\kappa = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$.*

– $\mathsf{PHF.Eval}(\kappa, X)$ *parses* $\kappa = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$ *and* $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ *computes and returns*

$$\mathrm{H}^{\mathsf{MG}}_\kappa(X) = h_0 \prod_{i=1}^{\ell} h_i^{x_i}$$

Essentially this function was already used, with an objective similar to ours in mind, in a construction from [25]. Here we provide a new use case and a useful abstraction of this function; also, we shed light on the properties of this function from different angles (i.e., for different values of m and n). In [25], it was implicitly proved that H^{MG} is a $(1, \mathsf{poly})$-PHF:

Theorem 5. *For any fixed polynomial* $q = q(k)$ *and group* $\mathbb{G}$ *with known order, the function* H^{MG} *is a* $(1, q)$*-programmable hash function with* $\gamma = 0$ *and* $\delta = 1/8(\ell+1)q$*.*

The proof builds upon the fact that $m = 1$ and does not scale in the m-component. With a completely different analysis, we can show that

Theorem 6. *For any group* $\mathbb{G}$ *with known order, the function* H^{MG} *is a* $(2, 1)$*-programmable hash function with* $\gamma = 0$ *and* $\delta = O(1/\ell)$*.*

Proof. We give only the intuition here. The full (and somewhat technical) proof appears in the full version. Consider the following algorithms:

– $\mathsf{PHF.TrapGen}(1^k, g, h)$ chooses $a_0, \ldots, a_\ell \in \{-1, 0, 1\}$ uniformly and independently, as well as random group exponents[3] $b_0, \ldots, b_\ell$. It sets $h_0 = g^{a_0 - 1} h^{b_0}$ and $h_i = g^{a_i} h^{b_i}$ for $1 \leq i \leq \ell$ and returns $\kappa = (h_0, \ldots, h_\ell)$ and $t = (a_0, b_0, \ldots, a_\ell, b_\ell)$.
– $\mathsf{PHF.TrapEval}(t, X)$ parses $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ and returns $a = a_0 - 1 + \sum_{i=1}^{\ell} a_i x_i$ and $b = b_0 + \sum_{i=1}^{\ell} b_i x_i$.

It is clear that this fulfills the syntactic and correctness requirements of Definition 1. Also, since the b_i are chosen independently and uniformly, so are the h_i, and the trapdoor keys indistinguishability requirement follows. It is more annoying to prove (3), and we will only give an intuition here. First, note that the $X_1, X_2, Z_1 \in \{0,1\}^\ell$ from (3) (for $m = 2$, $n = 1$) are independent of the a_i, since they are masked by the b_i in $h_i = g^{a_i} h^{b_i}$. Hence, if we view, e.g., X_1 as a subset of $[\ell]$ (where we define $i \in X_1$ iff the i-th component x_{1i} of X_1 is 1), then

$$a_{X_1} = a_0 - 1 + \sum_{i=1}^{\ell} a_i x_{1i} = -1 + a_0 + \sum_{i \in X_1} a_i$$

essentially[4] constitutes a random walk of length $|X_1| \leq \ell$. Theory says that it is likely that this random walk ends up with an a_{X_1} of small absolute value.

[3] If $|\mathbb{G}|$ is not known, this may only be possible approximately.

[4] Usually, random walks are formalized as a sum of independent values $a_i \in \{-1, 1\}$; for us, it is more convenient to assume $a_i \in \{-1, 0, 1\}$. However, this does not change things significantly.

That is, for any r with $|r| = O(\sqrt{\ell})$, the probability that $a_{X_1} = r$ is $\Theta(1/\sqrt{\ell})$. In particular, the probability for $a_{X_1} = 0$ is $\Theta(1/\sqrt{\ell})$. Now if X_1 and X_2 were disjoint and there was no a_0 in the sum, then a_{X_1} and a_{X_2} would be independent and we would get that $a_{X_1} = a_{X_2} = 0$ with probability $\Theta(1/\ell)$. But even if $X_1 \cap X_2 \neq \emptyset$, and taking into account a_0, we can conclude similarly by lower bounding the probability that $a_{X_1 \setminus X_2} = a_{X_2 \setminus X_1} = -a_{X_1 \cap X_2}$.

The additional requirement that $a_{Z_1} \neq 0$ with high probability is intuitively much more obvious, but also much harder to formally prove. First, without loss of generality, we can assume that $Z_1 \subseteq X_1 \cup X_2$, since otherwise, there is a "partial random walk" $a_{Z_1 \setminus (X_1 \cup X_2)}$ that contributes to a_{Z_1} but is independent of a_{X_1} and a_{X_2}. Hence, even when already assuming $a_{X_1} = a_{X_2} = 0$, a_{Z_1} still is sufficiently randomized to take a nonzero value with constant probability. Also, we can assume Z_1 not to "split" X_1 in the sense that $Z_1 \cap X_1 \in \{\emptyset, X_1\}$ (similarly for X_2). Otherwise, even assuming a fixed value of a_{X_1}, there is still some uncertainty about $a_{Z_1 \cap X_1}$ and hence about a_{Z_1} (in which case with some probability, a_{Z_1} does not equal any fixed value). The remaining cases can be handled with a similar "no-splitting" argument. However, note that the additional "-1" in the g-exponent of h_0 is essential: without it, picking X_1 and X_2 disjoint and setting $Z_1 = X_1 \cup X_2$ achieves $a_{Z_1} = a_{X_1} + a_{X_2} = 0$. A full proof is given in the full version.

Furthermore, using techniques from the proof of Theorem 6, we can asymptotically improve the bounds from Theorem 5 as follows (a proof can be found in the full version):

Theorem 7. *For any fixed polynomial $q = q(k)$ and group $\mathbb{G}$ with known order, the function H^{MG} is a $(1, q)$-programmable hash function with $\gamma = 0$ and $\delta = O(\frac{1}{q\sqrt{\ell}})$.*

One may wonder whether the scalability of H^{MG} with respect to m reaches further. Unfortunately, it does not (the proof for the following theorem appears in the full version):

Theorem 8. *Assume $\ell = \ell(k) \geq 2$. Say $|\mathbb{G}|$ is known and prime, and the discrete logarithm problem in $\mathbb{G}$ is hard. Then H^{MG} is not $(3, 1)$-programmable.*

If the group order $\mathbb{G}$ is not known (as will be the case in our upcoming RSA-based signature scheme), then it may not even be possible to sample group exponents uniformly. However, for the special case where $\mathbb{G} = \mathrm{QR}_N$ is the group of quadratic residues modulo $N = pq$ for safe distinct primes p and q, we can approximate a uniform exponent with a random element from $\mathbb{Z}_{N^2}$. In this case, the statistical distance between keys produced by $\mathsf{PHF.Gen}$ and those produced by $\mathsf{PHF.TrapGen}$ is smaller than $(\ell + 1)/N$. We get

Theorem 9. *For the group $\mathbb{G} = \mathrm{QR}_N$ of quadratic residues modulo $N = pq$ for safe distinct primes p and q, the function H^{MG} is $(\ell + 2)$-bounded $(1, q, (\ell + 1)/N, 1/8(\ell+1)q)$-programmable and also $(\ell+2)$-bounded $(2, 1, (\ell+1)/N, O(1/\ell))$-programmable.*

Similarly, one can show analogues of Theorem 8 and Theorem 3 for $\mathbb{G} = \mathrm{QR}_N$, only with the strong RSA assumption in place of the discrete log problem. We omit the details.

Randomized Programmable Hash Functions (RPHFs). In the full version we further generalize the notion of PHFs to randomized programmable hash functions (RPHFs). Briefly, RPHFs are PHFs whose evaluation is randomized, and where this randomness is added to the image (so that verification is possible). We show how to adapt the PHF definition to the randomized case, in a way suitable for the upcoming applications. We also give instantiations of RPHFs for parameters for which we do not know how to instantiate PHFs.

4 Applications of PHFs

4.1 Generic signatures from Bilinear Maps

Let $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p = |\mathbb{G}|, g, \hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T)$ be a pairing group. Let $n = n(k)$ and $\eta = \eta(k)$ be two arbitrary polynomials. Our signature scheme signs messages $m \in \{0,1\}^n$ using randomness $s \in \{0,1\}^\eta$.[5] Let a group hash function $\mathrm{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ with inputs from $\{0,1\}^n$ and outputs from $\mathbb{G}$ be given. We are ready to define our generic bilinear map signature scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$.

Key-Generation: Generate $\mathbb{PG}$ such that H can be used for the group $\mathbb{G}$. Generate a key for H via $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$. Pick a random index $x \in \mathbb{Z}_p$ and compute $X = g^x \in \mathbb{G}$. Return the public verification key $(\mathbb{PG}, X, \kappa)$ and the secret signing key x.

Signing: To sign $m \in \{0,1\}^n$, pick a random η-bit integer s and compute $y = \mathrm{H}_\kappa(m)^{\frac{1}{x+s}} \in \mathbb{G}$. The signature is the tuple $(s, y) \in \{0,1\}^\eta \times \mathbb{G}$.

Verification: To verify that $(s, y) \in \{0,1\}^\eta \times \mathbb{G}$ is a correct signature on a given message m, check that s is of length η, and that

$$\hat{e}(y, X \cdot g^s) = \hat{e}(H(m), g).$$

Theorem 10. *Let* H *be a* $(m, 1, \gamma, \delta)$*-programmable hash function. Let* $\mathcal{F}$ *be a* (t, q, ϵ)*-forger in the existential forgery under an adaptive chosen message attack experiment with* $\mathsf{SIG}_{\mathrm{BM}}$*. Then there exists an adversary* $\mathcal{A}$ *that* (t', ϵ')*-breaks the* q*-SDH assumption with* $t' \approx t$ *and*

$$\epsilon \leq \frac{q}{\delta} \cdot \epsilon' + \frac{q^{m+1}}{2^{m\eta - 1}} + \gamma\ .$$

[5] For signing arbitrary bitstrings, a collision resistant hash function $\mathsf{CR} : \{0,1\}^* \to \{0,1\}^n$ can be applied first. Due to the birthday paradox we choose $n = 2k$ when k bits of security are actually desired.

We remark that the scheme can also be instantiated in asymmetric pairing groups where the pairing is given by $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and $\mathbb{G}_1 \neq \mathbb{G}_2$. In that case we let the element y from the signature be in $\mathbb{G}_1$ such that y can be represented in 160 bits [6]. Also, in asymmetric pairings, verification can equivalently check if $\hat{e}(y, X) = \hat{e}(H(m) \cdot y^{-1/s}, g)$. This way we avoid any expensive exponentiation in $\mathbb{G}_2$ and verification time becomes roughly the same as in the Boneh-Boyen short signatures [6]. It can be verified that the following proof also holds in asymmetric pairing groups (assuming there exists an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$).

An efficiency comparison of the scheme instantiated with the $(2,1)$-PHF H^{MG} from Definition 4 appears in the full version.

Proof (Theorem 10). Let $\mathcal{F}$ be the adversary against the signature scheme. Throughout this proof, we assume that H is a (m, n, γ, δ)-programmable hash function. Furthermore, we fix some notation. Let m_i be the i-th query to the signing oracle and (s_i, y_i) denote the answer. Let m and (s, y) be the forgery output by the adversary. We introduce two types of forgers:
Type I: It always holds that $s = s_i$ for some i.
Type II: It always holds that $s \neq s_i$ for all i.
By $\mathcal{F}_1$ (resp., $\mathcal{F}_2$) we denote the forger who runs $\mathcal{F}$ but then only outputs the forgery if it is of type I (resp., type II). We now show that both types of forgers can be reduced to the $q+1$-SDH problem. Theorem 10 then follows by a standard hybrid argument.

Type I forgers.

Lemma 11. *Let $\mathcal{F}_1$ be a forger of type I that (t_1, q, ϵ_1)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-SDH assumption with $t' \approx t$ and*

$$\epsilon' \geq \frac{\delta}{q} \left(\epsilon_1 - \frac{q^{m+1}}{2^{m\eta}} - \gamma \right).$$

To prove the lemma we proceed in games. In the following, X_i denotes the probability for the adversary to successfully forge a signature in Game i.
Game 0. Let $\mathcal{F}_1$ be a type I forger that (t_1, q, ϵ_1)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. By definition, we have

$$\Pr[X_0] = \epsilon_1. \tag{4}$$

Game 1. We now generate trapdoor keys $(\kappa', t) \stackrel{\$}{\leftarrow} \mathsf{PHF.TrapGen}(1^k, g, h)$ for uniformly selected generators $g, h \in \mathbb{G}$ to generate a H-key for public verification key of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. By the programmability of H,

$$\Pr[X_1] \geq \Pr[X_0] - \gamma. \tag{5}$$

Game 2. Now we select the random values s_i used for answering signing queries not upon each signing query, but at the beginning of the experiment. Since the s_i were selected independently anyway, this change is only conceptual. Let $E = \bigcup_{i=1}^{q} \{s_i\}$ be the set of all s_i, and let $E^i = E \setminus \{i\}$. We also change the selection of the elements g, h used during $(\kappa', t) \stackrel{\$}{\leftarrow} \mathsf{PHF.TrapGen}(1^k, g, h)$ as follows. First, we uniformly choose $i^* \in [q]$ and a generator $\tilde{g} \in \mathbb{G}$. Define $p^*(\eta) = \prod_{t \in E^*}(\eta + t)$ and $p(\eta) = \prod_{t \in E}(\eta + t)$ and note that $\deg(p^*) \leq q - 1$ and $\deg(p) \leq q$. Hence the values $g = \tilde{g}^{p^*(x)}$, $h = \tilde{g}^{p(x)}$, and $X = g^x = \tilde{g}^{xp^*(x)}$ can be computed from $\tilde{g}, \tilde{g}^x, \ldots, \tilde{g}^{x^q}$. Here the index $x \in \mathbb{Z}_{|\mathbb{G}|}$ is the secret key of the scheme. We then set $E^* = E \setminus \{s_{i^*}\}$, $E^{*,i} = E^* \setminus \{i\}$, and

$$g = \tilde{g}^{p^*(x)} = \tilde{g}^{\prod_{t \in E^*}(x-t)}, \quad h = \tilde{g}^{p(x)} = \tilde{g}^{\prod_{t \in E}(x-t)}.$$

Note that we can compute $(x + s_i)$-th roots for $i \neq i^*$ from g and for all i from h. This change is purely conceptual:

$$\mathsf{Pr}[X_2] = \mathsf{Pr}[X_1]. \tag{6}$$

Observe also that i^* is independent of the adversary's view.

Game 3. In this game, we change the way signature requests from the adversary are answered. First, observe that the way we modified the generation of g and h in Game 2 implies that for any i with $s_i \neq s_{i^*}$, we have

$$\begin{aligned} y_i &= \mathrm{H}_{\kappa'}(m_i)^{\frac{1}{x+s_i}} = \left(g^{a_{m_i}} h^{b_{m_i}}\right)^{\frac{1}{x+s_i}} \\ &= \left(\tilde{g}^{a_{m_i} \prod_{t \in E^*}(x-t)} \tilde{g}^{b_{m_i} \prod_{t \in E}(x-t)}\right)^{\frac{1}{x+s_i}} = \tilde{g}^{a_{m_i} \prod_{t \in E^{*,i}}(x-t)} \tilde{g}^{b_{m_i} \prod_{t \in E^i}(x-t)} \end{aligned} \tag{7}$$

for $(a_{m_i}, b_{m_i}) \leftarrow \mathsf{PHF.TrapEval}(t, m_i)$. Hence for $i \neq i^*$, we can generate the signature (s_i, y_i) without explicitly knowing the secret key x, but instead using the right-hand side of (7) for computing y_i. Obviously, this change in computing signatures is only conceptual, and so

$$\mathsf{Pr}[X_3] = \mathsf{Pr}[X_2]. \tag{8}$$

Observe that i^* is still independent of the adversary's view.

Game 4. We now abort and raise event $\mathsf{abort_{coll}}$ if an s_i occurs more than m times, i.e., if there are pairwise distinct indices $i_1, \ldots, i_{m+1}$ with $s_{i_1} = \ldots = s_{i_{m+1}}$. There are $\binom{q}{m+1}$ such tuples $(i_1, \ldots, i_m)$. For each tuple, the probability for $s_{i_1} = \ldots = s_{i_{m+1}}$ is $1/2^{m\eta}$ A union bound shows that a $(m+1)$-wise collision occurs with probability at most

$$\mathsf{Pr}[\mathsf{abort_{coll}}] \leq \binom{q}{m+1} \frac{1}{2^{m\eta}} \leq \frac{q^{m+1}}{2^{m\eta}}.$$

Hence,

$$\mathsf{Pr}[X_4] \geq \mathsf{Pr}[X_3] - \mathsf{Pr}[\mathsf{abort_{coll}}] > \mathsf{Pr}[X_3] - \frac{q^{m+1}}{2^{m\eta}}. \tag{9}$$

Game 5. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.s}}$ if the adversary returns an $s \in E^*$, i.e., the adversary returns a forgery attempt (s, y) with $s = s_i$ for some i, but $s \neq s_{i^*}$. Since i^* is independent from the adversary's view, we have $\Pr[\mathsf{abort}_{\mathsf{bad.s}}] \leq 1 - 1/q$ for any choice of the s_i, so we get

$$\Pr[X_5] = \Pr[X_4 \wedge \neg\mathsf{abort}_{\mathsf{bad.s}}] \geq \frac{1}{q}\Pr[X_4]. \tag{10}$$

Game 6. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.a}}$ if there is an index i with $s_i = s_{i^*}$ but $a_{m_i} \neq 0$, or if $a_m = 0$ for the adversary's forgery message. In other words, we raise $\mathsf{abort}_{\mathsf{bad.a}}$ iff we do not have $a_{m_i} = 0$ for all i with $s_i = s_{i^*}$ and $a_{m_i} \neq 0$. Since we have limited the number of such i to m in Game 4, we can use the programmability of H. We hence have $\Pr[\mathsf{abort}_{\mathsf{bad.a}}] \leq 1 - \delta$ for any choice of the m_i and s_i, so we get

$$\Pr[X_6] \geq \Pr[X_5 \wedge \neg\mathsf{abort}_{\mathsf{bad.a}}] \geq \delta \cdot \Pr[X_5]. \tag{11}$$

Note that in Game 6, the experiment never really uses secret key x to generate signatures: to generate the y_i for $s_i \neq s_{i^*}$, we already use (7), which requires no x. But if $\mathsf{abort}_{\mathsf{bad.a}}$ does not occur, then $a_{m_i} = 0$ whenever $s_i = s_{i^*}$, so we can also use (7) to sign without knowing x. On the other hand, if $\mathsf{abort}_{\mathsf{bad.a}}$ does occur, we must abort anyway, so actually no signature is required.

This means that Game 6 does not use knowledge about the secret key x. On the other hand, the adversary in Game 6 produces (whenever X_6 happens, which implies $\neg\mathsf{abort}_{\mathsf{bad.a}}$ and $\neg\mathsf{abort}_{\mathsf{bad.s}}$) during a forgery

$$y = \mathrm{H}_{\kappa'}(m)^{1/(x+s)} = \left(\tilde{g}^{a_m \prod_{t \in E^*}(x+t)} \tilde{g}^{b_m \prod_{t \in E}(x+t)}\right)^{\frac{1}{x+s}} = \tilde{g}^{\frac{a_m p^*(x)}{x+s}} \tilde{g}^{b_m p^*(x)}.$$

From y and its knowledge about h and the s_i, the experiment can derive

$$y' = \left(\frac{y}{g^{b_m}}\right)^{1/a_m} = \tilde{g}^{\frac{p^*(x)}{x+s}}.$$

Since $\gcd(\eta + s, p^*(\eta)) = 1$ (where we interpret $\eta + s$ and $p^*(\eta)$ as polynomials in η), we can write $p^*(\eta)/(\eta + s) = p'(\eta) + q_0/(\eta + s)$ for some polynomial $p'(\eta)$ of degree at most $q - 2$ and some $q_0 \neq 0$. Again, we can compute $g' = \tilde{g}^{p'(x)}$. We finally obtain

$$y'' = (y'/g')^{1/q_0} = \left(\tilde{g}^{\frac{p(x)}{(x+s)} - p'(x)}\right)^{1/q_0} = \tilde{g}^{\frac{1}{x+s}}.$$

This means that the from the experiment performed in Game 6, we can construct an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-SDH assumption. $\mathcal{A}$'s running time t' is approximately t plus a small number of exponentiations, and $\mathcal{A}$ is successful whenever X_6 happens:

$$\epsilon' \geq \Pr[X_6]. \tag{12}$$

Putting (4-12) together yields Lemma 11.

Type II forgers.

Lemma 12. *Let $\mathcal{F}_2$ be a forger of type II that (t_1, q, ϵ_1)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-SDH assumption such that $t' \approx t$ and*

$$\epsilon' \geq \frac{\delta}{2}\left(\epsilon_2 - \gamma\right).$$

The difference is that a type II forger returns a valid signature (s, y) with $s \notin \{s_1, \ldots, s_q\}$. The idea of the reduction is that the simulation can be setup such that from this forgery an element $\tilde{g}^{\frac{1}{x+s}}$ can be computed which breaks the q-SDH assumption. The simulation of the signature queries is simular the one for type I forgers, where now we only have to use the $(1,1)$-programmability of H. A detailed proof is given in the full version.

4.2 Generic signatures from RSA

Let $\mathbb{G} = \mathrm{QR}_N$ be the group of quadratic residues modulo an RSA number $N = pq$, where p and q are safe primes. Let $n = n(k)$ and $\eta = \eta(k)$ be two polynomials. Let a group hash function $\mathrm{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ with inputs from $\{0,1\}^n$ and outputs from $\mathbb{G}$ be given. We are ready to define our generic RSA-based signature scheme $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$:

Key-Generation: Generate $N = pq$ for safe distinct primes $p, q \geq 2^{\eta+2}$, such that H can be used for the group $\mathbb{G} = \mathrm{QR}_N$. $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$. Return the public verification key (N, κ) and the secret signing key (p, q).

Signing: To sign $m \in \{0,1\}^n$, pick a random η-bit prime e and compute $y = \mathrm{H}_\kappa(m)^{1/e} \mod N$. The e-th root can be computed using p and q. The signature is the tuple $(e, y) \in \{0,1\}^\eta \times \mathbb{Z}_N$.

Verification: To verify that $(e, y) \in \{0,1\}^\eta \times \mathbb{Z}_N$ is a correct signature on a given message m, check that e is odd and of length η, and that $y^e = \mathrm{H}(m) \bmod N$.

Theorem 13. *Let* H *be a β-bounded $(m, 1, \gamma, \delta)$-programmable hash function for $\beta \leq 2^\eta$ and $m \geq 1$. Let $\mathcal{F}$ be a (t, q, ϵ)-forger in the existential forgery under an adaptive chosen message attack experiment with $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the strong RSA assumption with $t' \approx t$ and*

$$\epsilon = \Theta\left(\frac{q}{\delta} \cdot \epsilon'\right) + \frac{q^{m+1}(\eta+1)^m}{2^{m\eta-1}} + \gamma\,.$$

The proof is similar to the case of bilinear maps (Theorem 10). However, due to the fact that the group order is not known some technical difficulties arise which is the reason why we need the PHF to be β-bounded for some $\beta \leq 2^\eta$. The full formal proof appears in the full version.

Let us again consider the instantiation $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}^{\mathsf{MG}}]$ for the $(2,1)$-PHF H^{MG}. Plugging in the values from Theorem 9 the reduction from Theorem 13 leads to $\epsilon = \Theta(q\ell\epsilon') + \frac{q^3(\eta+1)^2}{2^{2\eta-1}}$. As explained in the introduction, for $q = 2^{30}$ and $k = 80$ bits we are now able to choose $\eta \approx 80$ bit primes.

References

1. Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494, Konstanz, Germany, May 11–15, 1997. Springer-Verlag, Berlin, Germany.
2. Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 216–233, Santa Barbara, CA, USA, August 21–25, 1994. Springer-Verlag, Berlin, Germany.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
4. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer-Verlag, Berlin, Germany.
5. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.
6. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.
7. 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.
8. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer-Verlag, Berlin, Germany.
9. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
10. Xavier Boyen. General ad hoc encryption from exponent inversion IBE. In *Advances in Cryptology—EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 394–411. Berlin: Springer-Verlag, 2007.
11. S. Brands. An efficient off-line electronic cash system based on the representation problem. Report CS-R9323, Centrum voor Wiskunde en Informatica, March 1993.
12. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289, Amalfi, Italy, September 12–13, 2002. Springer-Verlag, Berlin, Germany.
13. 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, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 127–141, Amsterdam, The Netherlands, April 13–15, 1988. Springer-Verlag, Berlin, Germany.
14. David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In Joan Feigenbaum,

editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 470–484, Santa Barbara, CA, USA, August 11–15, 1992. Springer-Verlag, Berlin, Germany.

15. Benoît Chevallier-Mames and Marc Joye. A practical and tightly secure signature scheme without hash function. In Masayuki Abe, editor, *CT-RSA 2007*, volume 4377 of *LNCS*, pages 339–356, San Francisco, CA, USA, February 5–9, 2007. Springer-Verlag, Berlin, Germany.
16. Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235, Santa Barbara, CA, USA, August 20–24, 2000. Springer-Verlag, Berlin, Germany.
17. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM CCS 99*, pages 46–51, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.
18. Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany.
19. Marc Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 116–129, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany.
20. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30, Santa Barbara, CA, USA, August 17–21, 1997. Springer-Verlag, Berlin, Germany.
21. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139, Prague, Czech Republic, May 2–6, 1999. Springer-Verlag, Berlin, Germany.
22. Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464, St. Petersburg, Russia, May 28 – June 1, 2006. Springer-Verlag, Berlin, Germany.
23. 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.
24. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.
25. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.
26. Huafei Zhu. New digital signature scheme attaining immunity to adaptive chosen-message attack. *Chinese Journal of Electronics*, 10(4):484–486, Oct 2001.