Programmable Hash Functions and Their Applications

Dennis Hofheinz[1] and Eike Kiltz[2]

[1] Institut für Kryptographie und Sicherheit, Karlsruhe Institute of Technology, Germany. Email: `Dennis.Hofheinz@kit.edu`.
[2] Fakultät für Mathematik, Ruhr-Universität Bochum. Email: `eike.kiltz@rub.de`.

Abstract. We introduce a new combinatorial 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 makes PHFs 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.

Keywords: Hash functions, digital signatures, standard model.

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 (user-specified) generators g and h from the group. This trapdoor information makes it possible to relate the output of the hash function H to g and h: for any input X, one obtains integers a_X and b_X such that the relation

$$\mathrm{H}(X) = g^{a_X} h^{b_X} \in \mathbb{G} \tag{1}$$

holds. For the PHF to be (m, n)-programmable we require that *for all* choices of $X_1, \ldots, X_m$ and $Z_1, \ldots, Z_n$ such that for all i, j it is true that $X_i \neq Z_j$, it holds that $a_{X_i} = 0$ but $a_{Z_j} \neq 0$, with significant probability:

$$\Pr\left[a_{X_1} = \ldots = a_{X_m} = 0 \wedge a_{Z_1}, \ldots, a_{Z_n} \neq 0\right] \geq 1/\,\mathsf{poly}\ . \tag{2}$$

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 do not depend on g (since $a_X = 0$). At the same time the hash images of all possible choices $Z_1, \ldots, Z_n$ of n (different) inputs do depend on g in a known way (since $a_Z \neq 0$).

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 [6] (which, in a sense, are ideally programmable hash functions), and has been used to derive proofs of the adaptive security of cryptosystems [7,14,12].

An (m, poly)-PHF is an (m, n)-PHF for all polynomials n. A (poly, m)-PHF is defined the same way. Note that, using this notation, a random oracle implies a $(\mathsf{poly}, 1)$-PHF.

Instantiations. As our central instantiation of a PHF we use the following function which was originally introduced by Chaum et. al. [23] 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 [23] it was also used in other constructions (e.g., [19,24,4,65]), relying on other useful properties beyond collision resistance. Specifically, in the analysis of his identity-based encryption scheme, Waters [65] 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}. Our analysis uses random walk techniques and is different from the one implicitly given in [65].

Variations. The concept of PHFs can be extended to randomized programmable hash functions (RPHFs). A RPHF is like a PHF whose input takes an additional parameter, the randomness. Our main constructions of a randomized hash functions are RH^{F} and RH^{L}. They are both $(1, 1)$-programmable and have *short* parameters. 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 $|a_X|$ from (1), for all X.

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 $a_Z \neq 0$ (so in particular $a_X \neq a_Z$). This implies $h = g^{a_Z/(b_X - b_Z)}$, revealing the discrete logarithm of h to the base g. (Note that already the weaker condition $a_X \neq a_Z$ is sufficient to imply collision resistance.)

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

$$\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}] : \quad sig = (\mathrm{H}(X)^{\frac{1}{x+s}}, s) \in \mathbb{G} \times \{0,1\}^\eta, \tag{3}$$

where s is interpreted as a random η-bit integer, and $x \in \mathbb{Z}_{|\mathbb{G}|}$ is the secret key. The signature can be verified with the help of the public key g, g^x and a bilinear map. This signature scheme can be seen as a generalization (resp. variation) of the schemes from [11,21,57]. 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 [11] holds, then the above signature scheme is unforgeable against chosen message attacks [39]. 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 [7], 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.)

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

$$\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]: \quad sig = (\mathrm{H}(X)^{1/e}, e) \in \mathbb{Z}_N \times \{0,1\}^\eta, \tag{4}$$

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 an $(m, 1)$-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 $\eta \approx 30 + 80/m$. Furthermore, our generic constructions explain signature schemes by Okamoto [57], Fischlin [33], variants of Zhu [66,67] and Camenisch and Lysyanskaya [21], and shed light why other proposals are not secure.

Other applications. BLS signatures [15] are an example of "full-domain hash" (FDH) signature schemes [6]. Using the properties of a $(m, 1)$-programmable hash function one can give a black-box reduction from m-time 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. Consequently, with a $(\mathsf{poly}, 1)$ PHF we obtain full unforgeability of full-domain signature schemes. Similarly, the Boneh-Franklin IBE scheme [13] can be proved secure under the Bilinear Diffie-Hellman assumption when instantiated with a $(\mathsf{poly}, 1)$-PHF. 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 [30].[3]

It is furthermore possible to reduce the security of Waters' IBE and signature scheme [65] 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 A Conceptual Perspective

We would like underline the importance of programmable hash functions as a *concept* for designing and analyzing cryptographic protocols in the Diffie-Hellman and RSA setting. The central idea is that one can partition the output of a hash function into two types of instances (c.f. (1) and (2)) that can be treated differently by a security reduction. This is reminiscent to what proofs in the random oracle model usually do (e.g., [7,14,12]) and hence PHFs offer a simple and abstract framework for designing and analyzing cryptographic protocols without explicitly relying on random oracles. 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 PRFs. To mention only a few examples, this ranges from collision-resistant hashing [23,4], digital signature schemes [11,65] (also in various flavors [57,61,46,8]), chosen-ciphertext secure encryption [18,49,43,44,17], identity-based encryption [9,10,51,22,2] to symmetric authentication [52]. In fact, besides a number of specific proofs, there seem to be only two generic techniques known to prove (Diffie-Hellman and RSA-based) cryptographic protocols in the standard model: the partitioning trick as abstracted in programmable hash functions and the recent *dual system* approach by Waters [64].

1.4 Short Signatures

Our main new applications of PHFs are short signatures in the standard model. We now discuss our results in more detail. We refer to [15,11] for applications of short signatures.

The birthday paradox and randomized signatures. A signature scheme $\mathsf{SIG}_{\mathrm{Fisch}}$ by Fischlin [33] (itself a variant of the RSA-based Cramer-Shoup signatures [28]) is defined as follows. The signature for a message

[3] We remark that the impossibility results from [30] do not imply that $(m, 1)$-programmable hash functions do not exist since they only rule out the possibility of proving the security of FDH signatures based on any assumption which is satisfied by random functions, thus it might still be possible to construct such objects using, say homomorphic properties.

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) 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 such 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). Hence, from two Fischlin signatures w.r.t. the same randomness e a signature can be computed (and hence the scheme can be broken). 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 $\eta > 80+60+8 = 148$ bits to immunize against this birthday attack. We remark that for a different reason, Fischlin' 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 (4), instantiated with a concrete (randomized) (1, 1)-PHF (RH^{F}). 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, $\mathrm{H}(X)^{1/e} = h^{b_X/e} = g^{b_X}$ 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 an $(m, 1)$-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 an $(m, 1)$-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 ratio 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)-PRF can have primes as small as $\eta = 80$ bits to be provably secure.[4] 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 (2, 1)-PRF only needs $\eta = 70$ bits of randomness to be provably secure.

Instantiations. 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. Concretely, since the complexity of finding a random η-bit prime with error 2^{-k} is $O(k\eta^4)$ we expect that, compared to the one by Fischlin, the signing algorithm of new scheme $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}^{\mathsf{MG}}]$ is roughly 16 times faster. Our generic bilinear construction instantiated with the RPHF RH^{F} explains the signature scheme by Fischlin [33]; instantiated with the the RPHF RH^{L} it explains a variant of the schemes by Zhu [66,67][5] and Camenisch and Lysyanskaya [21]. (Concretely, our variant uses a modified randomness space, see Appendix B for details.)

[4] A remark in [33, Sec. 2.3] concerning a stateless signature variant that can be securely instantiated with $\eta = 80$ bit primes turned out incorrect. Concretely, [33]-signatures are of the form (e, α, y) and satisfy $y^e = x h_1^\alpha h_2^{\alpha \oplus H(m)}$ for public h_1, h_2, x. In this, e is a 160-bit prime, and $\alpha \in \{0, 1\}^{160}$ is uniform. The remark in [33, Sec. 2.3] suggests to instead use a signature (e, α, y) with $y^e = x h_1^\alpha h_2^{\alpha \oplus H_1(m)} h_3^{\alpha \oplus H_2(m)}$ for $H(m) =: H_1(M) || H_2(M)$ and public h_1, h_2, h_3, x. This has the advantage that e and α can be chosen of size around 80 bits. It is claimed that the security proof of the original scheme can be adapted to this variant. However, the proof crucially uses that there is no collision among the e values used during the signing process, i.e., that no e occurs in more than one simulated signature. With 160-bit primes e, such a collision will occur only with small probability; but with 80-bit primes e, the probability will be in the order of $1/2^{40}$.

[5] The security proof given in [67] does not seem to be correct. Concretely, Zhu's signatures are of the form (e, α, y) and satisfy $y^e = h_0 h_1^\alpha h_2^{H(m)}$ for public h_0, h_1, h_2. In this, e is a random $2k$-bit prime, and $\alpha \in \{0, 1\}^\ell$ is uniform. However, in the security proof of a Type I adversary (for which $e = e_j \in \{e_1, \ldots, e_q\}$), one needs to argue that the simulated randomness α_j is uniformly distributed in $\{0, 1\}^\ell$. However, a close inspection of the used random

Scheme	Type	Signature Size	Key Size	Efficiency
Boneh-Boyen [11]	Bilinear	$\|\mathbb{G}\| + \|\mathbb{Z}_p\| = 320$	$2\|\mathbb{G}\| = 320$	$1 \times \mathsf{Exp}$
Okamoto [57] (=$\mathsf{SIG}_{\mathrm{BM}}[\mathrm{RH}^{\mathsf{L}}]$)	Bilinear	$\|\mathbb{G}\| + \|r\| + \|s\| = 480$	$4\|\mathbb{G}\| = 640$	$1 \times \mathsf{Exp}$
Ours: $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$	Bilinear	$\|\mathbb{G}\| + \|s\| = 230$	$(\ell + 2)\|\mathbb{G}\| = 26\mathrm{k}$	$1 \times \mathsf{Exp}$
Hohenberger-Waters [45]	RSA	$2 \times \|\mathbb{Z}_N\| = 2048$	$2 \times \|\mathbb{Z}_N\| = 2048$	$160 \times \mathsf{P}_{1024}$
Cramer-Shoup [28]	RSA	$2 \times \|\mathbb{Z}_N\| + \|e\| = 2208$	$3 \times \|\mathbb{Z}_N\| + \|e\| = 3232$	$1 \times \mathsf{P}_{160}$
Fischlin [33] (=$\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{RH}^{\mathsf{F}}]$)	RSA	$\|\mathbb{Z}_N\| + \|r\| + \|e\| = 1344$	$4 \times \|\mathbb{Z}_N\| = 4096$	$1 \times \mathsf{P}_{160}$
Ours: $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}^{\mathsf{MG}}]$	RSA	$\|\mathbb{Z}_N\| + \|e\| = 1104$	$(\ell + 1)\|\mathbb{Z}_N\| = 164\mathrm{k}$	$1 \times \mathsf{P}_{80}$

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. The efficiency column counts the dominant operations for signing. For Bilinear signatures this counts the number of exponentiations, for RSA signatures $k \times \mathsf{P}_\eta$ counts the number of random η-bit primes that need to be generated. We remark that the Hohenberger-Waters scheme relies only on the (non-strong) RSA assumption but its computational cost is incomparably higher.

In our comparison, we have also included the recent scheme of Hohenberger and Waters [45]. Their scheme has the benefit of relying only on the (non-strong) RSA assumption and having a compact verification key. However, their scheme requires a large number of primality tests and exponentiations during signing and verifying.

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 signatures scheme from Boneh-Boyen [11] and is only 70 bits larger than the random oracle BLS signatures. The signature scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{RH}^{\mathsf{L}}]$ is exactly the one proposed by Okamoto [57] (which was implicitly introduced in a group signature scheme [35]).

An obvious drawback of our constructions is the size of the public verification key since it includes the group hash key K. For example, for $\mathrm{H}^{\mathsf{MG}} : \{0,1\}^\ell \to \mathbb{G}$, K contains $\ell + 1$ group elements, where $\ell = 160$. In the bilinear case, that makes a verification key of $26k$ bits compared to 160 bits from [11]. 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 [54,38,22,17]. Furthermore, even for signatures in the random oracle model, sometimes a relatively large verification key is necessary [31].

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 [28,33], we also consider concrete signature size when additionally taking the non-tight security reduction into account. A rigorous comparison will be done in Section 7.

Related Signature Schemes. Our generic bilinear map signature scheme belongs to the class of "inversion-based" signature schemes originally proposed in [59] and first formally analyzed in [11]. The signature scheme from [57] can be viewed as a special case of our generic bilinear map signature scheme instantiated with a randomized PHF. Other related standard-model schemes can be found in [37,16]. We stress that our signatures derive from the above since the message does not appear in the denominator of the exponent. Our generic RSA signature scheme builds on the early work by Cramer and Shoup [28]. The signature schemes from [33] and variants of [21,66,67] can be viewed as a special case of our generic bilinear map signature scheme instantiated with a randomized PHF. Other standard-model RSA schemes are [36,20,26,56,40,29,48,60]. We remark that security proofs for strong RSA based signature schemes are quite subtle and several variants

variables shows that this is not the case (given the view of the adversary). Our variant $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{RH}^{\mathsf{L}}]$, as well as the scheme by Camenisch and Lysyanskaya [21] use larger randomness space to make the simulation work.

proposed in the literature contain flawed security proofs. As already explained in Footnote 4, a variant by Fischlin [33, Sec. 2.3] cannot be proved secure. Furthermore, the proof of a scheme proposed by Zhu [66,67] turned out to be incorrect (see Footnote 5) but a close variant with slightly larger randomness space (i.e., $\{0,1\}^L$ with $L = \ell + k$ instead of $L = \ell$) can be proved secure using our framework.

1.5 Dedicated vs. Programmable Hash Functions

As argued before, random oracles [6] can be viewed as excellent programmable hash functions. For common applications such as full-domain hash signatures or OAEP, one usually instantiates the random oracle with a fixed, dedicated hash function (such as SHA1 [62]), Therefore, one may ask the question if such concrete hash functions (when used as keyed hash functions) can serve as good programmable hash functions. More concretely, is SHA1 an (m, n)-PRF for parameters $m, n \geq 1$?

Even though it seems hard to actually disprove, our intuition says that this is very likely not the case. In fact, one of the key design maxims of hash functions like SHA1 is to *destroy* all algebraic structure. In contrast, the definition of programmable hash functions requires that *there is* a relation over an algebraic structure. (I.e., we require that $\mathrm{H}(X) = g^{a_X} h^{b_X}$ over the group $\mathbb{G}$.) In that sense programmable hash functions formalize an obvious weakness in the random oracle methodology: security proofs making in the random oracle model often use a property of the hash function that is commonly avoided by hash function's designers. Therefore, we do not recommend to use dedicated hash functions as a PHF.

1.6 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. Another interesting direction is to find instantiations of PHFs from different assumptions. For instance, the ideas in [22,2,17] seem conceptually close to programmable hash functions in lattices.

We leave it as an open problem to prove or disprove the standard-model existence of $(\mathsf{poly}, 1)$-PHFs. (Note that a positive result would imply a security proof for FDH signatures like [7,15]). Moreover, we are asking for a concrete construction of a bounded $(m, 1)$-PHF for $m > 2$.[6] For example, a $(3, 1)$-PHF could be used to shrink the signature size of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ to ≈ 215 bits; a bounded $(5, 1)$-PHF would make it possible to shrink the size of the prime in $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$ to roughly $\eta = 60$ bits and make signing roughly as efficient as RSA full-domain hash[7] (with the drawback of a larger public-key). 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 [65].

2 Preliminaries

2.1 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.

6 We remark that an earlier version of this paper contained a generalization of RH^{F} to a randomized $(m, 1)$-PHF for any $m \geq 2$. However, for our applications it did not turn out to be useful. Since for $m \geq 2$ it is not sufficiently bounded (it is only $2^{\ell m}$-bounded), it does not lead to more efficient RSA-based signatures. In the bilinear case, the instantiations with this RPHF are all less efficient than Boneh-Boyen signatures.

7 For $\eta \approx 60$ a full exponentiation modulo a 1024-bit integer become roughly as expensive as finding a random η-bit prime.

2.2 Digital signatures

A digital signature scheme SIG consists of three PPT algorithms. The key generation algorithm inputs a security parameter (in unary representation) and 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 property.

We recall the definition for unforgeability against chosen-message attacks (UF-CMA), played between a challenger and a forger $\mathcal{F}$:

1. On input of the security parameter k, 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 (in the security parameter k).

2.3 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 $\hat{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. See [11] 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 is a uniform generator of $\mathbb{G}$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. We require that in $\mathbb{PG}$ the q-SDH [11] assumption holds meaning that no adversary can (t, ϵ) break the q-SDH problem for a polynomial t and non-negligible ϵ.

2.4 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 primes P and Q of $k/2$ bits. (A safe prime is a prime number of the form $2P' + 1$, where P' is also a prime.) In our later constructions, we will also use QR_N, 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 [3,34] 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

3.1 Definitions

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 $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$ is generated by the key generation algorithm $\mathsf{PHF.Gen}$. This key K can then be used for the deterministic evaluation algorithm $\mathsf{PHF.Eval}$ to evaluate H via $y \leftarrow \mathsf{PHF.Eval}(K, X) \in \mathbb{G}$ for any $X \in \{0,1\}^\ell$. We write $\mathrm{H}_K(X) = \mathsf{PHF.Eval}(K, X)$.

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 $g, h \in \mathbb{G}$, the trapdoor key generation $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ produces a key K' along with a trapdoor t. Moreover, $(a_X, b_X) \leftarrow \mathsf{PHF.TrapEval}(t, X)$ produces integers a_X and b_X for any $X \in \{0,1\}^\ell$.*

Correctness: *We demand $\mathrm{H}_{K'}(X) = \mathsf{PHF.Eval}(K', X) = g^{a_X} h^{b_X}$ for all generators $g, h \in \mathbb{G}$ and all possible $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$, 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 $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$ and $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$, the keys K and K' are statistically γ-close: $K \overset{\gamma}{\equiv} K'$.*

Well-distributed logarithms: *For all generators $g, h \in \mathbb{G}$ and all possible K' in the range of (the first 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*

$$\Pr\left[a_{X_1} = \ldots = a_{X_m} = 0 \quad \wedge \quad a_{Z_1}, \ldots, a_{Z_n} \neq 0\right] \geq \delta, \tag{5}$$

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

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.*

We remark that the requirement of the statistically close trapdoor keys is somewhat reminiscent to the concept of "lossy trapdoor functions" [58]. 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.*

3.2 Instantiations

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 (c, poly) or (poly, c)-programmable hash function, for any constant $c > 0$: 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 [27]: the random oracle on some input X is defined to be as $\mathcal{O}(X) := g^{\Delta_X \cdot \tilde{a}_X} \cdot h^{(1-\Delta_X)\tilde{b}_X}$, where Δ_X is a random biased coin with $\Pr[\Delta_X = 1] := 1/(2q(k))$ and $\tilde{a}_X$ and $\tilde{b}_X$ are uniform values from $\mathbb{Z}_{|\mathbb{G}|}$. Then (5) is fulfilled with probability $(1 - 1/(2q(k)))^{q(k)} \cdot (1/(2q(k)))^c \geq 1/(4q(k))^c$, meaning $\mathcal{O}$ is a (poly, c)-programmable hash function.)

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

Definition 3 (Multi-Generator PHF). *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 and independently sampled $K = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$.*

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

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

Essentially this function was already used, with an objective similar to ours in mind, in a construction from [65]. 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 [65], it was implicitly proved that H^{MG} is a $(1, \mathsf{poly})$-PHF:

Theorem 4. *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 5. *For any group $\mathbb{G}$ with known order, the function H^{MG} is a $(2, 1)$-programmable hash function with $\gamma = 0$ and $\delta = \Theta(1/\ell)$.*

Proof. We give only the intuition here and postpone the full (and somewhat technical) proof to Appendix A.1. Consider the following algorithms:

- $\mathsf{PHF.TrapGen}(1^k, g, h)$ sets $a_0 = -1$ and chooses uniformly and independently $a_1, \ldots, a_\ell \in \{-1, 0, 1\}$ and random group exponents[8] $b_0, \ldots, b_\ell$. It sets $h_i = g^{a_i} h^{b_i}$ for $0 \le i \le \ell$ and returns $K = (h_0, \ldots, h_\ell)$ 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 $a = a_0 + \sum_{i=1}^{\ell} a_i x_i$ 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 challenging to prove (5) (for $m = 2$, $n = 1$), i.e., that for all strings X_1, X_2 and $Z_1 \notin \{X_1, X_2\}$, we have that

$$\mathsf{Pr}\left[a_{X_1} = a_{X_2} = 0 \wedge a_{Z_1} \neq 0\right] = \Theta(1/\ell) \; . \tag{6}$$

We will only give an intuition here. First, note that the X_1, X_2, Z_1 are independent of the a_i, since they are masked by the b_i in $h_i = g^{a_i} h^{b_i}$. If we view 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 the value

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

essentially[9] constitutes a *random walk* of length $|X_1| + 1 \le \ell + 1$. Theory says that it is likely that this random walk ends up with an a_{X_1} of small absolute value. That is, for any d with $|d| = O(\sqrt{\ell})$, the probability that $a_{X_1} = d$ 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 from (6) that $a_{Z_1} \neq 0$ 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,

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

[9] 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.

even when already assuming $a_{X_1} = a_{X_2} = 0$, a_{Z_1} still is sufficiently randomized to take a non-zero 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 fixed "$a_0 = -1$" 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 Appendix A.1.

Using techniques from the proof of Theorem 5, we can asymptotically improve the bounds from Theorem 4 as follows (a proof can be found in Appendix A):

Theorem 6. *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 is in Appendix A):

Theorem 7. *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 $\mathbb{Z}_{N^2}$. (See, e.g., [28].) 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 the following theorem.

Theorem 8. *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 $O(q\ell)$-bounded $(1, q, (\ell+1)/N, 1/8(\ell+1)q)$-programmable as well as $O(\ell)$-bounded $(2, 1, (\ell+1)/N, O(1/\ell))$-programmable.*

As is to be expected, one can show that also in case $\mathbb{G} = \mathrm{QR}_N$, the function H^{MG} is *not* $(3, 1)$-programmable.

3.3 Randomized Programmable Hash Functions (RPHFs)

In Appendix B 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 Basic applications of PHFs

4.1 Collision resistant hashing

As a warm-up, we can show the natural result that any (non-trivially) programmable hash function is collision-resistant.

Theorem 9. *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.*

Proof. Fix PPT algorithms $\mathsf{PHF.TrapGen}$ and $\mathsf{PHF.TrapEval}$. To show H's collision-resistance, assume an adversary $\mathcal{A}$ that outputs a collision with non-negligible probability with keys $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$. Now by the key closeness of Definition 1, $\mathcal{A}$ will also do so with keys K' from $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$, for any g, h. Any collision $\mathrm{H}_{K'}(X) = \mathrm{H}_{K'}(X')$ with $X \neq X'$ gives rise to an equation

$$g^a h^b = \mathrm{H}_{K'}(X) = \mathrm{H}_{K'}(X') = g^{a'} h^{b'},$$

where $(a, b) \leftarrow \mathsf{PHF.TrapEval}(t, X)$ and $(a', b') \leftarrow \mathsf{PHF.TrapEval}(t, X')$. (5) states that with non-negligible probability, we have $a = 0$ and $a' \neq 0$, in which case we can compute $\mathrm{dlog}_h(g) = (b - b')/a' \mod |\mathbb{G}|$.

Similarly (using Lemma 14), one can show that for a PHF for $\mathbb{G} = \mathrm{QR}_N$, $(1, 1)$-programmability implies collision-resistance under the strong RSA assumption. We omit the details.

4.2 Other applications

As already discussed in the introduction, PHFs have other applications.

- A $(\mathsf{poly}, 1)$-PHF is sufficient to instantiate the hash function used in full-domain hash signatures like BLS signatures or RSA-FDH. A fair number of other protocols (e.g., the Boneh/Frankin IBE scheme [13]) are based on the same "full-domain hash" properties of the hash function. Unfortunately, we do not know if $(\mathsf{poly}, 1)$-PHFs do exist, or not. Similarly, a $(m, 1)$-PHF is sufficient to instantiate the hash function used in full-domain hash signatures like BLS signatures or RSA-FDH and show that they are secure m-time signatures.
- A $(1, \mathsf{poly})$-PHF is sufficient to instantiate the "hash function" used in Waters' IBE and signature scheme [65]. In fact, the $(1, \mathsf{poly})$-PHF H^{MG} is the original hash function Waters used in his IBE scheme. Our new bound from Theorem 6 can be used to improve the bound in the security reduction of Waters' IBE and signature scheme. We expect that the same improvements can be achieved for schemes based on Waters' IBE, e.g., [1,5,18,50,53].

5 Generic signatures from Bilinear Maps

5.1 Construction

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 $s \in \{0,1\}^\eta$.[10] 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 $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$. Pick a random index $x \in \mathbb{Z}_p^*$ and compute $X = g^x \in \mathbb{G}$. Return the public verification key $(\mathbb{PG}, X, K)$ and the secret signing key x.

Signing: To sign $m \in \{0,1\}^n$, pick a random η-bit integer s and compute $y = \mathrm{H}_K(m)^{\frac{1}{x+s}} \in \mathbb{G}$. 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

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

[10] 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.

Theorem 10. *Let* H *be an* $(m, 1, \gamma, \delta)$*-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*

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

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 $\mathbb{G}_1 \neq \mathbb{G}_2$. We use MNT curves [55] such that the element $y \in \mathbb{G}_1$ from the signature can be represented in 160 bits. (See [11] for more details.) Also, in asymmetric pairings, verification can equivalently check if $\hat{e}(y, X) = \hat{e}(\mathrm{H}_K(m) \cdot y^{-1/s}, g)$. 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 [11]. It can be verified that the following proof also holds in asymmetric pairing groups. (Note that the security assumption also has to be adapted to symmetric q-SDH assumption which is given $g_1, g_1^x, \ldots, g_1^{(x^q)}, g_2, g_2^x$, it is hard to find a pair $(c, g_1^{1/(x+c)})$.)

An efficiency comparison of the scheme instantiated with the $(2, 1)$-PHF H^{MG} from Definition 3 is done in Section 7.

5.2 Proof of Theorem 10

Let $\mathcal{F}$ be the adversary against the signature scheme. Throughout this proof, we assume that H is a $(m, 1, \gamma, \delta)$-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.

Both reductions rely on a trick from [11] that given a q-SDH instance $\tilde{g}, \tilde{g}^x, \ldots, \tilde{g}^{x^q}$, one can efficiently compute g, g^x, together with q random solved instances $(g^{1/(x+s_i)}, s_i)$. A new instance of the form $(g^{1/(x+s)}, s)$ for $s \notin \{s_1, \ldots, s_q\}$, however, can be used to break the q-SDH assumption. For Type II forgers this idea can be applied more or less directly. For Type I forgers it may happen that there is a m-collision in the simulated randomness, i.e, we have $s = s_{i_1} = \ldots s_{i_m}$, and one has to use the properties of the $(m, 1)$-PHF to be able to simulate the maximal m signatures of the form $(\mathrm{H}(m_{i_j})^{1/(x+s)}, s)$, while using the forger's output $\mathrm{H}(m)^{1/(x+s)}$ to break the q-SDH assumption.

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*

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

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

$$\Pr[X_0] = \epsilon_1. \tag{7}$$

Game 1. We now use the trapdoor key generation $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ 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,

$$\Pr[X_1] \geq \Pr[X_0] - \gamma. \tag{8}$$

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 $E^i = E \setminus \{s_i\}$. We also change the selection of the elements g, h used during $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ as follows. First, we uniformly choose $i^* \in [q]$ and a generator $\tilde{g} \in \mathbb{G}$. Define $E^* = E \setminus \{s_{i^*}\}$ and $E^{*,i} = E^* \setminus \{s_i\}$. Further, define the polynomials $p^*(\eta) = \prod_{t \in E^*}(\eta + t)$ 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

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

Note that we can compute $(x+s_i)$-th roots for $i \neq i^*$ from g and for all i from h. Unless we are in the unlucky case that g or h are not generators (which can only happens if $p(x) = 0$) this change is purely conceptual:

$$\Pr[X_2] \geq \Pr[X_1] - \frac{q}{p}. \tag{9}$$

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

$$\begin{aligned} y_i = \mathrm{H}_{K'}(m_i)^{\frac{1}{x+s_i}} &= \left(g^{a_{m_i}} h^{b_{m_i}}\right)^{\frac{1}{x+s_i}} \\ &= \left(\tilde{g}^{a_{m_i}\prod_{t \in E^*}(x+t)} \tilde{g}^{b_{m_i}\prod_{t \in E}(x+t)}\right)^{\frac{1}{x+s_i}} = \tilde{g}^{a_{m_i}\prod_{t \in E^{*,i}}(x+t)} \tilde{g}^{b_{m_i}\prod_{t \in E^i}(x+t)} \end{aligned} \tag{10}$$

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 (10) for computing y_i. Obviously, this change in computing signatures is only conceptual, and so

$$\Pr[X_3] = \Pr[X_2]. \tag{11}$$

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 an $(m+1)$-wise collision occurs with probability at most

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

Hence,

$$\Pr[X_4] \geq \Pr[X_3] - \Pr[\mathsf{abort_{coll}}] > \Pr[X_3] - \frac{q^{m+1}}{2^{m\eta}}. \tag{12}$$

Game 5. We now abort and raise event $\mathsf{abort_{bad.s}}$ 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 $\Pr[\mathsf{abort_{bad.s}}] \leq 1 - 1/q$ for any choice of the s_i, so we get

$$\Pr[X_5] = \Pr[X_4 \wedge \neg\mathsf{abort_{bad.s}}] \geq \frac{1}{q}\Pr[X_4]. \tag{13}$$

Game 6. We now abort and raise event $\mathsf{abort_{bad.a}}$ 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 $\mathsf{abort_{bad.a}}$ iff we do not have $a_{m_i} = 0$ for all i with $s_i = s_{i^*}$ and $a_m \neq 0$. Since we have limited the number of such i to m in Game 4, we can use the programmability of H. We hence have $\mathsf{Pr}[\mathsf{abort_{bad.a}}] \leq 1 - \delta$ for any choice of the m_i and s_i, so we get

$$\mathsf{Pr}[X_6] \geq \mathsf{Pr}[X_5 \wedge \neg \mathsf{abort_{bad.a}}] \geq \delta \cdot \mathsf{Pr}[X_5]. \tag{14}$$

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 (10), which requires no x. But if $\mathsf{abort_{bad.a}}$ does not occur, then $a_{m_i} = 0$ whenever $s_i = s_{i^*}$, so we can also use (10) to sign without knowing x. On the other hand, if $\mathsf{abort_{bad.a}}$ 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 $\neg\mathsf{abort_{bad.a}}$ and $\neg\mathsf{abort_{bad.s}}$) during a forgery

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

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

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

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:

$$\epsilon' \geq \mathsf{Pr}[X_6]. \tag{15}$$

Putting (7-15) together yields Lemma 11.

Type II forgers

Lemma 12. *Let $\mathcal{F}_2$ be a forger of type II that (t_2, q, ϵ_2)-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 and an adversary $\mathcal{A}^*$ that (t'', ϵ'')-breaks the discrete logarithm problem in $\mathbb{G}$ such that $t', t'' \approx t_2$ and*

$$\epsilon' + \epsilon'' \geq \delta \cdot (\epsilon_2 - \gamma).$$

Note that the discrete logarithm problem is at least as hard as the q-SDH problem, so for Theorem 10, we can assume $\epsilon' \geq \epsilon''$ without loss of generality.

For the proof, we again proceed in games. The proof is very similar to the proof for type I forgers, so we will be brief where similarities occur.

Game 0. Let $\mathcal{F}_2$ be a type II forger that (t_2, q, ϵ_2)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. By definition, we have

$$\mathsf{Pr}[X_0] = \epsilon_2. \tag{16}$$

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

$$\mathsf{Pr}[X_1] \geq \mathsf{Pr}[X_0] - \gamma. \tag{17}$$

Game 2. Now we select the used randomness s_i used for answering signing queries at the beginning of the experiment and set $E = \bigcup_{i=1}^{q}\{s_i\}$. We select the elements g, h passed to $\mathsf{PHF.TrapGen}(1^k, g, h)$ as follows: We uniformly choose a generator $\tilde{g} \in \mathbb{G}$. Define the polynomial $p(\eta) = \prod_{t \in E}(\eta + t)$ and note that $\deg(p) \leq q$. Hence the values $g = \tilde{g}^{p(x)}$ and $X = g^x = \tilde{g}^{xp(x)}$ can be computed from $\tilde{g}, \tilde{g}^x, \ldots, \tilde{g}^{x^{q+1}}$. We choose $c \in \mathbb{Z}_{|\mathbb{G}|}$ uniformly and set

$$g = \tilde{g}^{p(x)}, \quad h = \tilde{g}^{cp(x)}.$$

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

$$\mathsf{Pr}[X_2] = \mathsf{Pr}[X_1]. \tag{18}$$

Game 3. We answer all signature requests from the adversary as in Game 3 of the proof of Lemma 11. That is, we use the way that g and h are chosen to avoid having to compute the $(x + s_i)$th root. This change is only conceptual, and we have

$$\mathsf{Pr}[X_3] = \mathsf{Pr}[X_2]. \tag{19}$$

Game 4. We now abort and raise event $\mathsf{abort}_{\mathsf{log}}$ if $a_m + c \cdot b_m = 0 \mod |\mathbb{G}|$ for the adversary's forged message m. Since we chose c as a uniform exponent and only pass g and $h = g^c$ (but no further information about c) to adversary and $\mathsf{PHF.TrapGen}$, these algorithms break a discrete logarithm problem. In particular, we can construct a suitable (t'', ϵ'')-attacker $\mathcal{A}^*$ on the discrete logarithm problem in $\mathbb{G}$ that takes g^c as input and computes $c = -a_m/b_m \mod |\mathbb{G}|$. This adversary achieves

$$\mathsf{Pr}[X_4] \geq \mathsf{Pr}[X_3 \wedge \neg\mathsf{abort}_{\mathsf{log}}] \geq \mathsf{Pr}[X_3] - \epsilon''. \tag{20}$$

Game 5. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.a}}$ if a_m (obtained from $\mathsf{PHF.TrapEval}(t, m)$) is zero for the adversary's forgery message m. The programmability of H directly implies

$$\mathsf{Pr}[X_5] \geq \mathsf{Pr}[X_4 \wedge \neg\mathsf{abort}_{\mathsf{bad.a}}] \geq \delta \cdot \mathsf{Pr}[X_4]. \tag{21}$$

Now from Game 5, we can now construct an adversary $\mathcal{A}$ on the $(q+1)$-SDH assumption. $\mathcal{A}$ takes inputs $\tilde{g}, \tilde{g}^x, \ldots, \tilde{g}^{x^{q+1}}$ and simulates Game 5 with adversary $\mathcal{F}_2$. $\mathcal{A}$ uses its inputs as if it was selected by the experiment; note that in Game 5, the secret key x is not used anymore. Now whenever $\mathcal{F}_2$ outputs a forgery y with

$$y = \left(g^{a_m} h^{b_m}\right)^{\frac{1}{x+s}} = \left(\tilde{g}^{(a_m + c \cdot b_m)\prod_{t \in E}(x+t)}\right)^{\frac{1}{x+s}}.$$

Since we have $a_m + c \cdot b_m \neq 0 \mod |\mathbb{G}|$, we can compute a nontrivial root of the challenge $\tilde{g}$. Therefore, from

$$y' = y^{\frac{1}{ca_m + db_m}} = \tilde{g}^{\frac{p(x)}{x+s}}$$

one can compute $\tilde{g}^{1/(x+s)}$, like in the proof of Lemma 11. Putting (16-21) together (and using that $\delta \leq 1$) yields Lemma 12.

6 Generic signatures from RSA

6.1 Construction

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 $\mathbb{G} = \mathrm{QR}_N$. $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^k)$. Return the public verification key (N, K) and the secret signing key (P, Q).

Signing: To sign $m \in \{0,1\}^n$, pick a random η-bit prime e and compute $y = \mathrm{H}_K(m)^{1/e} \mod N$. 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 $y^e = \mathrm{H}(m) \bmod N$. It is not necessary to check specifically that e is a prime.

Theorem 13. *Let* H *be a β-bounded $(m, 1, \gamma, \delta)$-programmable hash function for bound $\beta \leq 2^\eta$ 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*

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

The proof is similar to the case of bilinear maps (Theorem 10).

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 8 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.

6.2 Proof of Theorem 13

We first state the following simple lemma due to [41].

Lemma 14. *Given $x, z \in \mathbb{Z}_n^*$, along with $a, b \in \mathbb{Z}$, such that $x^a = z^b$, one can efficiently compute $\tilde{x} \in \mathbb{Z}_n^*$ such that $\tilde{x} = z^{\frac{\gcd(a,b)}{a}}$.*

To prove this lemma one can use the extended Euclidean algorithm to compute integers f, g such that $bf + ag = \gcd(a, b)$. One can check that $\tilde{x} := x^f z^g$ satisfies the above equation.

Now let $\mathcal{F}$ be the adversary against the signature scheme. Throughout this proof, we assume that H is a $(m, 1, \gamma, \delta)$-programmable hash function. Furthermore, we fix some notation. Let m_i the ith query to the signing oracle an (e_i, y_i) denote the answer. Let m and (e, y) be the forgery output by the adversary. We introduce two types of forgers:

Type I: It always holds that $e = e_i$ for some i.

Type II: It always holds that $e \neq e_i$ 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 strong RSA problem. Theorem 13 then follows by a standard hybrid argument.

Similar to the q-SDH case, both reductions rely on the standard trick [28] that given an RSA instance $N = pq$ and $\tilde{g} \in \mathrm{QR}_N$, one can efficiently compute $g \in \mathrm{QR}_N$, together with q random solved instances $(g^{1/e_i}, e_i)$, for random primes e_i. A new instance of the form $(g^{1/e}, e)$ for $e \notin \{e_1, \ldots, e_q\}$, however, can be used to break the strong RSA assumption. For Type II forgers this idea can be applied more or less directly. For Type I forgers it may happen that there is a m-collision in the simulated random primes, i.e, we have $e = e_{i_1} = \ldots e_{i_m}$, and one has to use the properties of the $(m,1)$-PHF to be able to simulate the maximal m signatures of the form $(\mathrm{H}(m_{i_j})^{1/e}, e)$, while using the forger's output $\mathrm{H}(m)^{1/e}$ to break the strong RSA assumption.

Type I forgers

Lemma 15. *Let $\mathcal{F}_1$ be a forger of type I that (t_1, q, ϵ_1)-breaks the existential unforgeability of* $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$*. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the strong RSA assumption with $t' \approx t$ and*

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

To prove the lemma we proceed in games.

Game 0. Let $\mathcal{F}_1$ be a type I forger that (t_1, q, ϵ_1)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. By definition, we have

$$\Pr[X_0] = \epsilon_1. \tag{22}$$

Game 1. We now use the trapdoor key generation $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ for uniformly selected generators $g, h \in \mathrm{QR}_N$ to generate a H-key for the public verification key of $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. By the programmability of H,

$$\Pr[X_1] \geq \Pr[X_0] - \gamma. \tag{23}$$

Game 2. Now we select the used primes e_i used for answering signing queries not upon each signing query, but at the beginning of the experiment. Since the e_i were selected independently anyway, this change is only conceptual. Let $E = \bigcup_{i=1}^{q} e_i$ be the set of all e_i, and let $E^i = E \setminus \{i\}$. We also change the selection of the elements g, h used during $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ as follows. First, we uniformly choose $i^* \in [q]$ and generators $\tilde{g} \in \mathbb{Z}_N^*, \tilde{h} \in \mathrm{QR}_N$. We then set $E^* = E \setminus \{e_{i^*}\}$, $E^{*,i} = E^* \setminus \{e_i\}$, and

$$g = \tilde{g}^{2 \prod_{x \in E^*} x}, \quad h = \tilde{h}^{\prod_{x \in E} x}.$$

Note that we can extract an e_i-th root for $i \neq i^*$ from g and for all i from h. Unless none of the e_i divides $|\mathbb{G}|$, the induced distribution on g and h is the same as in Game 1. Since $|\mathbb{G}| = |\mathrm{QR}_N| = P'Q'$ for primes $P' = (P-1)/2$ and $Q' = (Q-1)/2$, and we assumed that $P, Q \geq 2^{\eta+2}$, however, we have that e_i does not divide $|\mathbb{G}|$ (for all i).

$$\Pr[X_2] = \Pr[X_1]. \tag{24}$$

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 $e_i \neq e_{i^*}$, we have that y_i can be written as

$$\mathrm{H}_{K'}(m_i)^{1/e_i} = \left(g^{a_{m_i}} h^{b_{m_i}}\right)^{1/e_i} = \left(\tilde{g}^{2a_{m_i} \prod_{x \in E^*} x} \tilde{h}^{b_{m_i} \prod_{x \in E} x}\right)^{1/e_i} = \tilde{g}^{2a_{m_i} \prod_{x \in E^{*,i}} x} \tilde{h}^{b_{m_i} \prod_{x \in E^i} x}.$$

for $(a_{m_i}, b_{m_i}) \leftarrow \mathsf{PHF.TrapEval}(t, m_i)$. Hence for $i \neq i^*$, we can generate the signature (e_i, y_i) without explicit exponent inversion, but instead using this alternative presentation of y_i. Obviously, this change in computing signatures is only conceptual, and so

$$\Pr[X_3] = \Pr[X_2]. \tag{25}$$

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

Game 4. We now abort and raise event $\mathsf{abort_{coll}}$ if an e_i occurs more than m times, i.e., if there are pairwise distinct indices $i_1, \ldots, i_{m+1}$ with $e_{i_1} = \ldots = e_{i_{m+1}}$. There are $\binom{q}{m+1}$ such tuples $(i_1, \ldots, i_m)$. For each tuple, the probability for $e_{i_1} = \ldots = e_{i_{m+1}}$ is $1/\mathcal{P}^m$, where $\mathcal{P}$ denotes the number of primes[11] of length η. Since $\mathcal{P} > 2^\eta / 3(\eta+1)\log 2$ (see, e.g., [63, Theorem 5.7]), a union bound shows that an $(m+1)$-wise collision occurs with probability at most

$$\Pr[\mathsf{abort_{coll}}] \leq \binom{q}{m+1} \left(\frac{3(\eta+1)\log 2}{2^\eta}\right)^m \leq \frac{q^{m+1}(\eta+1)^m}{2^{m\eta}} \cdot \frac{(3\log 2)^m}{(m+1)!} < \frac{q^{m+1}(\eta+1)^m}{2^{m\eta-1}}.$$

[11] For simplicity, we assume a uniform distribution among all primes of length η.

Hence,

$$\mathsf{Pr}\left[X_4\right] \geq \mathsf{Pr}\left[X_3\right] - \mathsf{Pr}\left[\mathsf{abort}_{\mathsf{coll}}\right] > \mathsf{Pr}\left[X_3\right] - \frac{q^{m+1}(\eta+1)^m}{2^{m\eta-1}}. \tag{26}$$

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

$$\mathsf{Pr}\left[X_5\right] = \mathsf{Pr}\left[X_4 \wedge \neg\mathsf{abort}_{\mathsf{bad.e}}\right] \geq \frac{1}{q}\mathsf{Pr}\left[X_4\right]. \tag{27}$$

Game 6. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.a}}$ if there is an index i with $e_i = e_{i^*}$ 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 $e_i = e_{i^*}$ and $a_m \neq 0$. Since we have limited the number of such i to m in Game 4, we can use the programmability of H. We hence have $\mathsf{Pr}\left[\mathsf{abort}_{\mathsf{bad.a}}\right] \leq 1 - \delta$ for any choice of the m_i and e_i, so we get

$$\mathsf{Pr}\left[X_6\right] \geq \mathsf{Pr}\left[X_5 \wedge \neg\mathsf{abort}_{\mathsf{bad.a}}\right] \geq \delta \cdot \mathsf{Pr}\left[X_5\right]. \tag{28}$$

Note that in Game 6, the experiment never really needs to invert exponents to generate signatures: to generate the y_i for $e_i \neq e_{i^*}$, we already use the method of Game 3, which requires no inversion. But if $\mathsf{abort}_{\mathsf{bad.a}}$ does not occur, then $a_{m_i} = 0$ whenever $e_i = e_{i^*}$, so we can also use that method to sign without inversion. 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 factorization of N. On the other hand, the adversary in Game 6 produces (whenever X_6 happens, which implies $\neg\mathsf{abort}_{\mathsf{bad.a}}$ and $\neg\mathsf{abort}_{\mathsf{bad.e}}$) during a forgery

$$y = (\mathrm{H}_{K'}(m))^{1/e} = \left(\tilde{g}^{2a_m \prod_{x \in E^*} x} \tilde{h}^{b_m \prod_{x \in E} x}\right)^{1/e} = \tilde{g}^{\frac{2a_m \prod_{x \in E^*} x}{e}} \cdot \tilde{h}^{b_m \prod_{x \in E^*} x}.$$

From y and its knowledge about $\tilde{h}$, and the e_i, the experiment can derive

$$y' = \frac{y}{\tilde{h}^{b_m \prod_{x \in E^*} x}} = \tilde{g}^{\frac{2a_m \prod_{x \in E^*} x}{e}}.$$

We have $\gcd(2a_m \prod_{x \in E^*} x, e) = 1$ because e is larger than $|a_m|$ by H's boundedness, so that Lemma 14 finally allows to obtain $y'' = \tilde{g}^{1/e}$. Since $\tilde{g}$ was chosen initially independently and uniformly from $\mathbb{Z}_N^*$, this means that the from the experiment performed in Game 6, we can construct an adversary $\mathcal{A}$ that (t', ϵ')-breaks the strong RSA 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:

$$\epsilon' \geq \mathsf{Pr}\left[X_6\right]. \tag{29}$$

Putting (22-29) together yields Lemma 15.

Type II forgers

Lemma 16. *Let $\mathcal{F}_2$ be a forger of type II that (t_1, q, ϵ_1)-breaks the existential unforgeability of* $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. *Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the strong RSA assumption with $t' \approx t$ and*

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

Again we proceed in games. The proof is very similar to the proof for type I forgers, so we will be brief where similarities occur.

Game 0. Let $\mathcal{F}_2$ be a type II forger that (t_2, q, ϵ_2)-breaks the existential unforgeability of $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. By definition, we have

$$\mathsf{Pr}\left[X_0\right] = \epsilon_2. \tag{30}$$

Game 1. We now use the trapdoor key generation $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^k, g, h)$ for uniformly selected generators $g, h \in \mathrm{QR}_N$ to generate a H-key for public verification key of $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. By the programmability of H,

$$\Pr[X_1] \geq \Pr[X_0] - \gamma. \tag{31}$$

Game 2. Now we select the used primes e_i used for answering signing queries at the beginning of the experiment and set $E = \bigcup_{i=1}^{q} e_i$. We select the elements g, h passed to $\mathsf{PHF.TrapGen}(1^k, g, h)$ as follows: we choose $\tilde{g} \in \mathbb{Z}_N^*$ and $c \in \mathbb{Z}_{N^2}$ uniformly and set

$$g = \tilde{g}^{2\prod_{x \in E} x}, \quad h = g^c = \tilde{g}^{2c\prod_{x \in E} x}.$$

Note that we can extract an e_i-th root from g and h for all i. These change is purely conceptual:

$$\Pr[X_2] = \Pr[X_1]. \tag{32}$$

Game 3. We answer all signature requests from the adversary as in Game 3 of the proof of Lemma 15. That is, we use the way that g and h are chosen to avoid having to invert exponents. This change is only conceptual, and we have

$$\Pr[X_3] = \Pr[X_2]. \tag{33}$$

Game 4. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.e}}$ if e divides $a_m + c \cdot b_m$ over the integers. Recall that $|\mathbb{G}| = |\mathrm{QR}_N| = p'q'$ for primes p', q' with $N = (2p'+1)(2q'+1)$. Recall also that c is chosen uniformly from $\mathbb{Z}_{N^2}$, so we can write $c = c_1 + c_2|\mathbb{G}|$ with $0 \leq c_1 < |\mathbb{G}|$. Note that c_2 is statistically $1/N$-close to being uniformly distributed over $\{0, \ldots, \lfloor \frac{N^2-1}{p'q'} \rfloor\}$ and independent of c_1. However, the only information about c released to the adversary and the $\mathsf{PHF.TrapGen}$ algorithm is $h = g^c$ and hence $c_1 = c \mod |\mathbb{G}|$.

We would like to find necessary conditions for $\mathsf{abort}_{\mathsf{bad.e}}$. To this end, let $d = \gcd(b_m, e)$. We first claim that for $\mathsf{abort}_{\mathsf{bad.e}}$, it is necessary that $d \neq e$. For contradiction, assume $d = e$. Then $e | b_m$ by definition of d. Since $|a_m| < e$ by H's boundedness, we also have $e \nmid a_m + c \cdot b_m$. Taken together this implies that e does not divide $a_m + c \cdot b_m$, and hence we have $\neg\mathsf{abort}_{\mathsf{bad.e}}$. Next, we show that for $\mathsf{abort}_{\mathsf{bad.e}}$, we need to have $d | a_m$. Again, assume $d \nmid a_m$ for contradiction. Then, $d | c \cdot b_m$ and $d | e$ by definition of d. Hence, $e \nmid a_m + c \cdot b_m$, and again $\neg\mathsf{abort}_{\mathsf{bad.e}}$ is implied.

So we can assume $d \neq e$ and $d | a_m$ without loss of generality in our analysis of $\mathsf{abort}_{\mathsf{bad.e}}$. Then $\mathsf{abort}_{\mathsf{bad.e}}$ is equivalent to

$$a_m + c \cdot b_m = 0 \mod e \Leftrightarrow \frac{a_m}{d} + (c_1 + c_2|\mathbb{G}|)\frac{b_m}{d} = 0 \mod \frac{e}{d} \Leftrightarrow c_2 = -|\mathbb{G}|^{-1}\left(\frac{a_m}{d}\left(\frac{b_m}{d}\right)^{-1} + c_1\right),$$

which occurs with probability at most $1/3 + 1/N$ due to the distribution of c_2. (Note that $|\mathbb{G}| = p'q'$ is invertible modulo e/d since $|p'|, |q'|$ are prime and longer than e, and b_m/d is invertible by construction of d.) We get

$$\Pr[X_4] \geq \Pr[X_3 \wedge \neg\mathsf{abort}_{\mathsf{bad.e}}] \geq \left(\frac{2}{3} - \frac{1}{N}\right)\Pr[X_3] \geq \frac{1}{2} \cdot \Pr[X_3]. \tag{34}$$

Game 5. We now abort and raise event $\mathsf{abort}_{\mathsf{bad.a}}$ if a_m (obtained from $\mathsf{PHF.TrapEval}(t, m)$) is zero for the adversary's forgery message m. The programmability of H directly implies

$$\Pr[X_5] \geq \Pr[X_4 \wedge \neg\mathsf{abort}_{\mathsf{bad.a}}] \geq \delta\Pr[X_4]. \tag{35}$$

Now from Game 5, we can now construct an adversary $\mathcal{A}$ on the strong RSA assumption. $\mathcal{A}$ takes inputs N and $\tilde{g} \in \mathbb{Z}_N^*$ and simulates Game 5 with adversary $\mathcal{F}_2$. $\mathcal{A}$ uses $\tilde{g}$ as well as N just as if it was selected by the experiment; note that in Game 5, no inversion of exponents is necessary anymore. Now whenever $\mathcal{F}_2$ outputs a forgery, this implies in particular that no $\mathsf{abort}_{\mathsf{bad.e}}$ event was raised and we have

$$f := \gcd(a_m + c \cdot b_m, e) = \gcd(2(a_m + c \cdot b_m) \prod_{x \in E} x, e) < e,$$

so that we can use Lemma 14 to compute $\tilde{g}^{e/f}$ from every successful forgery

$$y = \left(g^{a_m} h^{b_m}\right)^{1/e} = \left(\tilde{g}^{2(a_m + c \cdot b_m) \prod_{x \in E} x}\right)^{1/e}.$$

Hence we can compute a nontrivial root of the challenge $\tilde{g}$ and thus break the strong RSA assumption:

$$\epsilon' \geq \Pr[X_5]. \tag{36}$$

Putting (30-36) together yields Lemma 16 and completes the proof of Theorem 13

7 Signature Sizes

In this section we compute the concrete size of our bilinear maps signatures $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ when instantiated with the multi-generator PHF H^{MG} and compare it to the size of known schemes. A similar comparison can be made for our RSA signatures $\mathsf{SIG}_{\mathrm{RSA}}[\mathrm{H}]$. Here we only focus on signature sizes. Let us stress again that the key sizes of our signature schemes are considerably larger compared to other schemes.

7.1 Concrete Security

This subsection follows the concrete security approach by Bellare and Ristenpart [5], which in turn builds upon the concrete success measure from [42].

For any adversary $\mathcal{A}$ running in time $\mathbf{T}(\mathcal{A})$ and gaining advantage ϵ we define the *success ratio of* $\mathcal{A}$ to be $\mathbf{SR}(\mathcal{A}) := \epsilon / \mathbf{T}(\mathcal{A})$. The ratio of $\mathcal{A}$'s advantage to its running time provides a measure of the efficiency of the adversary. Generally speaking, to resist an adversary with success ration $\mathbf{SR}(\mathcal{A})$, a scheme should choose its security parameter (bits of security) such that $\mathbf{SR}(\mathcal{A}) \leq 2^{-k}$ (with respect to the best known attack).

Security of the q-DH Assumption. We consider Cheon's attacks against the q-DH assumption [25] over groups of prime order p. The main result of [25] is that there exists an adversary $\mathcal{P}$ such that

$$\mathbf{SR}(\mathcal{P}) = \frac{\epsilon_{\mathcal{P}}}{\mathbf{T}(\mathcal{P})} = \frac{\mathbf{T}^2(\mathcal{P}) \cdot q}{p \cdot \mathbf{T}(\mathcal{P})} = \Omega(\sqrt{q/p}).$$

For our analysis we make the assumption that $\sqrt{q/p}$ is the maximal success ratio of an adversary against the q-DH problem, i.e., that

$$\mathbf{SR}(\mathcal{B}) \leq \sqrt{q/p}, \tag{37}$$

for all possible adversaries $\mathcal{B}$. (We note that $\mathbf{SR}(\mathcal{P}) = \Theta(\sqrt{q/p})$ matches the generic lower bounds from [11].)

Our signature scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$. For our setting, we consider an uf-cma adversary $\mathcal{A}$ against the signature scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ that makes q signing queries, runs in time $\mathbf{T}(\mathcal{A})$, and has advantage ϵ. We can relate the success ratio of $\mathcal{A}$ to the success ration of the adversary $\mathcal{B}$ against the q-DH problem from our reduction. Namely, applying Theorem 10 we have that

$$\mathbf{SR}(\mathcal{A}) \leq \frac{1}{\mathbf{T}(\mathcal{B})} \cdot \left(\frac{q}{\delta} \cdot \epsilon' + \frac{q^{m+1}}{2^{\eta m}}\right) = \frac{q}{\delta} \cdot \mathbf{SR}(\mathcal{B}) + \frac{q^{m+1}}{2^{\eta m}} \cdot \frac{1}{\mathbf{T}(\mathcal{B})} \leq \frac{q}{\delta} \cdot \mathbf{SR}(\mathcal{B}) + \frac{q^m}{2^{\eta m}}. \tag{38}$$

We want that the signature scheme has k *bit security*, i.e., that $\mathbf{SR}(\mathcal{A}) \leq 2^{-k}$. Combining this with (37) and (38) we obtain

$$\mathbf{SR}(\mathcal{A}) \leq \frac{q}{\delta} \cdot \sqrt{q/p} + \frac{q^m}{2^{\eta m}} \leq 2^{-k+1}. \tag{39}$$

(To simplify the upcoming equations we only opt for $k-1$ bit security.) We are interested in the minimal choice of the group order p and the (bit-)length η of the randomness such that the above equation holds. Clearly, (39) is satisfied if both,

$$\eta \geq \log q + \frac{k}{m} \tag{40}$$

and

$$\log p \geq 2k + 3\log q - 2\log\delta \tag{41}$$

hold.

The signature scheme by Boneh and Boyen. The security reduction for Boneh/Boyen signatures to the q-DH assumption is tight, i.e., $\mathbf{SR}(\mathcal{A}) \approx \mathbf{SR}(\mathcal{B}) \leq \sqrt{p/q}$ which, for k bit security, again has to be bounded by 2^{-k}. Therefore we need to chose p such that

$$\log p \geq 2k + 2\log q\ . \tag{42}$$

Note the size of the randomness η in the Boneh/Boyen signatures is always fixed, i.e., $\eta = \log p$.

7.2 Concrete comparison

We make a comparison for $k = 80$ bits. For concreteness we consider the instantiation $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$ for the hash function H^{MG} from Definition 3. By Theorem 5 this is a the $(2,1)$-PHF with $\delta = \frac{1}{c\ell} \approx 2^{-3\log k}$ and $\gamma = 0$. We will perform two types of comparisons.

Ignoring increase of the group. First, as it is common in the literature [28,33,11], we ignore the penalty imposed on the group size due to the non-tight reduction and Cheon's attack. That is, ignoring (41) and (42) we always chose $\log p = 2k$ bits, independent of the number of signature queries an adversary can make. This is reasonable when one views a security reduction as an asymptotic indicator of security. However, the bound from (40) on the randomness η cannot be ignored since, as shown in the introduction, this may lead to an actual attack on the signature scheme. The signatures of $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ consist of one group element plus η bit randomness, the signatures of $\mathsf{SIG}_{\mathrm{BB}}$ of one group element plus randomness which consists of one element from $\mathbb{Z}_p$. On special Bilinear Maps with the representation of one element in $|\mathbb{G}|$ takes exactly $\log p = 2k$ bits [11], we obtain

$$|\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]| = \log p + \eta = 2k + \log q + \frac{k}{m}, \qquad |\mathsf{SIG}_{\mathrm{BB}}| = 2\log p = 4k\ .$$

For different choices of k and q the resulting signature sizes are given in the top two rows of Table 2. For example, for $k = 80$ bits security, it seems realistic to assume that an adversary makes maximal $q \in \{2^{20}, 2^{30}, 2^{40}\}$ signature queries.

Taking the increase of the group into account. We now compute the signature sizes when also taking the increase of the underlying group size into account. Using (41) and (40) for $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ and (42) for $\mathsf{SIG}_{\mathrm{BB}}$ we obtain

$$|\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]| = \log p + \eta = k(2 + \frac{1}{m}) + 6\log k + 4\log q, \qquad |\mathsf{SIG}_{\mathrm{BB}}| = 2\log p = 4k + 4\log q\ .$$

For different choices of k and q the signature sizes are given in the bottom two rows of Table 2.

Online/offline signature generation. We mention, however, that the Boneh-Boyen signature has an interesting online/offline property. Namely, almost all of the computational work of signing can be outsourced into a precomputation phase. Later, when the messages to be signed are known, using the precomputation results, signatures can be prepared extremely efficiently. Our scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}]$ does not seem to inherit this property.

Scheme	Signature size								
	$k=80$			$k=128$			$k=256$		
	$q=2^{20}$	$q=2^{30}$	$q=2^{40}$	$q=2^{32}$	$q=2^{48}$	$q=2^{64}$	$q=2^{64}$	$q=2^{96}$	$q=2^{128}$
Fixed Group Size									
Boneh-Boyen [11]	320	320	320	512	512	512	1024	1024	1024
Ours: $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$	220	230	240	352	368	384	704	736	768
Variable Group Size									
Boneh-Boyen [11]	400	440	480	640	704	768	2304	2432	2560
Ours: $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{H}^{\mathsf{MG}}]$	316	356	396	490	554	618	944	1072	1200

Table 2. Recommended signature sizes of different schemes. The top two rows give the sizes when ignoring the increase of the group due to the non-tight generic bounds and the bottom two rows take the latter into account.

Acknowledgements

The authors would like to thank the anonymous reviewers for helpful suggestions, and Jonathan Katz and Wei Gao for pointing out a number of smaller mistakes in an earlier version. The second author was funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

1. Michel Abdalla, Dario Catalano, Alex Dent, John Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311, Venice, Italy, July 10–14, 2006. Springer, Berlin, Germany.
2. 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, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.
3. Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Germany.
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, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Berlin, Germany.
5. Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
7. 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, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.
8. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *Public Key Cryptography*, pages ???–???, 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, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.
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, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

11. 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.
12. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.
13. Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
14. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer, Berlin, Germany.
15. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
16. 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.
17. 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, Paris, France, May 26–28, 2010. Springer, Berlin, Germany.
18. 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, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press.
19. S. Brands. An efficient off-line electronic cash system based on the representation problem. Report CS-R9323, Centrum voor Wiskunde en Informatica, March 1993.
20. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, September 12–13, 2002. Springer, Berlin, Germany.
21. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
22. 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, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.
23. 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, Amsterdam, The Netherlands, April 13–15, 1988. Springer, Berlin, Germany.
24. David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Berlin, Germany.
25. Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.
26. 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, San Francisco, CA, USA, February 5–9, 2007. Springer, Berlin, Germany.
27. Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.
28. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.
29. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 256–271, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Berlin, Germany.

30. 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, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.
31. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.
32. William Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1, 3rd Edition*. Wiley, 1968.
33. 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, Miami, USA, January 6–8, 2003. Springer, Berlin, Germany.
34. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.
35. Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 05: 10th Australasian Conference on Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 455–467, Brisbane, Queensland, Australia, July 4–6, 2005. Springer, Berlin, Germany.
36. 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, Prague, Czech Republic, May 2–6, 1999. Springer, Berlin, Germany.
37. Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.
38. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
39. 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.
40. Jens Groth. Cryptography in subgroups of $\mathbb{Z}_n$. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 50–65, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Germany.
41. Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, May 25–27, 1988. Springer, Berlin, Germany.
42. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
43. 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, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany.
44. 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, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.
45. 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, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
46. Qiong Huang and Duncan S. Wong. New constructions of convertible undeniable signature schemes without random oracles. Cryptology ePrint Archive, Report 2009/517, 2009. `http://eprint.iacr.org/`.
47. Barry D. Hughes. *Random Walks and Random Environments: Volume 1: Random Walks*. Oxford University Press, 1995.
48. Marc Joye. How (not) to design strong-RSA signatures. *Designs, Codes and Cryptography*, pages ???–???, 2011.
49. 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, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.
50. Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *ACISP 2006*, volume 4058 of *LNCS*, pages 336–347. Springer-Verlag, 2006.
51. 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.

52. Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In *EUROCRYPT*, pages ???–???, 2011.
53. Eike Kiltz and Yevgeniy Vahlis. CCA2 secure IBE: Standard model efficiency through authenticated symmetric encryption. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 221–238, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.
54. 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, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
55. Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, 2001.
56. David Naccache, David Pointcheval, and Jacques Stern. Twin signatures: An alternative to the hash-and-sign paradigm. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 20–27, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.
57. 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, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.
58. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
59. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.
60. Sven Schäge. Tight proofs for signature schemes without random oracles. In *EUROCRYPT*, pages ???–???, 2011.
61. 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, Tenerife, Canary Islands, Spain, January 25–28, 2010. Springer, Berlin, Germany.
62. Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, April 1995.
63. Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.
64. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
65. 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, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.
66. Huafei Zhu. New digital signature scheme attaining immunity to adaptive chosen-message attack. *Chinese Journal of Electronics*, 10(4):484–486, Oct 2001.
67. Huafei Zhu. A formal proof of zhus signature scheme. Cryptology ePrint Archive, Report 2003/155, 2003. `http://eprint.iacr.org/`.

A Proofs from Section 3

A.1 Random walks and the full proof of Theorem 5

The goal of this section is to prove Theorem 5. As indicated, this requires some work; in particular, we need some theory about random walks. For a thorough introduction, we refer to [47,32]. Here, we will only use (variations of) some elementary results. For self-containment, we give some basic proofs below.

The first theorem summarizes some elementary facts about one-dimensional random walks:

Theorem 17 (Random walks with $\{-1,1\}$-steps). *Let $\mu \in \mathbb{N}_{>0}$ and $a'_1, \ldots, a'_\mu \in \{-1,1\}$ be independently and uniformly distributed random variables. For $i \in \mathbb{Z}$, let*

$$p'_\mu(i) := \mathsf{Pr}\left[\sum_{j=1}^{\mu} a'_j = i\right],$$

where the probability is over the a'_i. *Then*

$$p'_\mu(i) = 0 \qquad \text{if } i \not\equiv \mu \bmod 2, \tag{43}$$

$$p'_\mu(-i) = p'_\mu(i) \qquad \text{for } i \in \mathbb{Z}, \tag{44}$$

$$p'_\mu(i+2) \le p'_\mu(i) \qquad \text{for } i \in \mathbb{N}_0,\ i \equiv \mu \bmod 2, \tag{45}$$

$$p'_{\mu+2}(0) < p'_\mu(0). \tag{46}$$

Furthermore, there exists $\Lambda' > 0$ *and, for every* $c > 0$, *also* $\lambda'_c > 0$, *such that for all* i *with* $i \equiv \mu \bmod 2$ *and* $|i| \le c\sqrt{\mu}$,

$$\lambda'_c \le p'_\mu(i)\sqrt{\mu} \le \Lambda'. \tag{47}$$

Proof. (43) and (44) follow from the definition, and (45) is easiest seen by writing

$$p'_\mu(i) = 2^{-\mu}\binom{\mu}{(\mu+i)/2} = 2^{-\mu}\frac{\mu!}{(\mu/2+i/2)!(\mu/2-i/2)!}$$

(for $i \in \mathbb{N}_0$, $i \equiv \mu \bmod 2$) for $p'_\mu(i)$ and $p'_\mu(i+2)$ and subtracting them. (46) follows by observing that

$$p'_{\mu+2}(0) = \frac{p'_\mu(-2) + 2p'_\mu(0) + p'_\mu(2)}{4} \overset{(44)}{=} \frac{p'_\mu(2) + p'_\mu(0)}{2} \overset{(45)}{\le} p'_\mu(0).$$

To see the upper bound in (47), we may assume $i \ge 0$ because of (44). Note that

$$p'_\mu(i) \overset{(45)}{\le} p'_\mu(i \bmod 2) = 2^{-\mu}\binom{\mu}{\lceil \mu/2 \rceil} = 2^{-\mu}\frac{\mu!}{\lceil \mu/2\rceil! \lfloor \mu/2 \rfloor!} \overset{(*)}{=} \Theta(1/\sqrt{\mu}),$$

where $(*)$ uses Stirling's approximation. (Θ is asymptotic in μ.) For the lower bound, $m' := \lfloor c\sqrt{\mu} \rfloor$, and $m := m' - ((\mu - m') \bmod 2)$, so m is the largest possible value for i in (47). Now

$$\begin{aligned}
p'_\mu(i) &\overset{(45)}{\ge} p'_\mu(m) = 2^{-\mu}\binom{\mu}{(\mu+m)/2} = 2^{-\mu}\frac{\mu!}{((\mu+m)/2)!((\mu-m)/2)!} \\
&\overset{(*)}{=} \Theta\left(\sqrt{\frac{\mu}{(\mu+m)(\mu-m)} \cdot \frac{\mu^{2\mu}}{(\mu+m)^{\mu+m}(\mu-m)^{\mu-m}}}\right) = \Theta\left(\sqrt{\frac{1}{\mu} \cdot \frac{1}{(1-\frac{m^2}{\mu^2})^{\mu-m}} \cdot \frac{1}{(1+\frac{m}{\mu})^{2m}}}\right) \\
&\ge \Theta\left(\frac{1}{\sqrt{\mu}} \cdot \frac{1}{(1+\frac{c}{\sqrt{\mu}})^{2c\sqrt{\mu}}}\right) = \Theta\left(\frac{1}{\sqrt{\mu}} \cdot e^{-2c^2}\right) = \Theta\left(\frac{1}{\sqrt{\mu}}\right)
\end{aligned}$$

as desired, where $(*)$ denotes again Stirling's approximation.

However, for our purposes, it is more useful to allow zero-steps, since this avoids (43).

Theorem 18 (Random walks with $\{-1,0,1\}$-steps). *Let* $\mu \in \mathbb{N}_{>0}$ *and* $a_1, \ldots, a_\mu \in \{-1,0,1\}$ *be independently and uniformly distributed random variables. For* $i \in \mathbb{Z}$, *let*

$$p_\mu(i) := \mathsf{Pr}\left[\sum_{j=1}^{\mu} a_j = i\right],$$

where the probability is over the a_i. *Then*

$$p_\mu(-i) = p_\mu(i) \qquad \text{for } i \in \mathbb{Z}, \tag{48}$$

$$p_\mu(i+1) \le p_\mu(i) \qquad \text{for } i \in \mathbb{N}_0. \tag{49}$$

Furthermore, there exists $\Lambda > 0$ and, for every $c > 0$, also $\lambda_c > 0$, such that for all i with $|i| \leq c\sqrt{\mu}$,

$$\lambda_c \leq p_\mu(i)\sqrt{\mu} \leq \Lambda. \tag{50}$$

Also,

$$\frac{\lambda_c}{\Lambda} p_\mu(i_1) \leq p_\mu(i_2) \leq \frac{\Lambda}{\lambda_c} p_\mu(i_1) \tag{51}$$

for arbitrary i_1, i_2 with $|i_1|, |i_2| \leq c\sqrt{\mu}$. Finally, for every $c > 0$, there exists $\Gamma_c > 0$ independent of μ such that

$$\mathsf{Pr}\left[\left|\sum_{j=1}^{\mu} a_j\right| \leq c\sqrt{\mu}\right] \geq \Gamma_c. \tag{52}$$

Proof. (48) follows from the definition. (49) can be seen by induction on μ. For $\mu = 1$, (49) is clear. Now assume (49) for μ and, for $i \geq 0$, consider

$$p_{\mu+1}(i) = \frac{p_\mu(i-1) + p_\mu(i) + p_\mu(i+1)}{3} \geq \frac{p_\mu(i) + p_\mu(i+1) + p_\mu(i+2)}{3} = p_{\mu+1}(i+1).$$

This shows (49) for $\mu + 1$, and hence in general. Next, we prove the upper bound in (50). To this end, let $n_0 := |\{j \mid a_j = 0\}|$ be the number of zeros among the a_j. Clearly, the expectation value of n_0 is $\mu/3$. Hence, using Hoeffding's inequality, we first obtain

$$\mathsf{Pr}\left[n_0 \geq \mu/2\right] \leq e^{-\mu/18}. \tag{53}$$

We get

$$\begin{aligned}
p_\mu(i) \overset{(49)}{\leq} p_\mu(0) = \mathsf{Pr}\left[\sum_{a_j \neq 0} a_j = 0\right] &= \sum_{i=0}^{\lfloor \mu/2 \rfloor} \mathsf{Pr}\left[\sum_{a_j \neq 0} a_j = 0 \mid n_0 = \mu - 2i\right] \mathsf{Pr}\left[n_0 = \mu - 2i\right] \\
&= \sum_{i=0}^{\lfloor \mu/2 \rfloor} p'_{2i}(0) \mathsf{Pr}\left[n_0 = \mu - 2i\right] \overset{(53)}{\leq} e^{-\mu/18} + \sum_{i=\lfloor \mu/4 \rfloor}^{\lfloor \mu/2 \rfloor} p'_{2i}(0) \mathsf{Pr}\left[n_0 = \mu - 2i\right] \\
&\overset{(46)}{\leq} e^{-\mu/18} + \sum_{i=\lfloor \mu/4 \rfloor}^{\lfloor \mu/2 \rfloor} p'_{2\lfloor \mu/4 \rfloor}(0) \mathsf{Pr}\left[n_0 = \mu - 2i\right] \leq e^{-\mu/18} + p'_{2\lfloor \mu/4 \rfloor}(0) \overset{(47)}{=} \Theta(1/\sqrt{\mu}).
\end{aligned}$$

This provides an upper bound Λ on $p_\mu(i)/\sqrt{\mu}$. To derive a lower bound, assume a fixed $c > 0$, and write $m := 2\lceil c\sqrt{\mu}/2 \rceil$ (i.e., m is the smallest even upper bound on $c\sqrt{\mu}$). We get:

$$\begin{aligned}
p_\mu(i) \overset{(49)}{\geq} p_\mu(m) = \mathsf{Pr}\left[\sum_{a_j \neq 0} a_j = m\right] &= \sum_{i=0}^{\lfloor \mu/2 \rfloor} \mathsf{Pr}\left[\sum_{a_j \neq 0} a_j = m \mid n_0 = \mu - 2i\right] \mathsf{Pr}\left[n_0 = \mu - 2i\right] \\
&= \sum_{i=0}^{\lfloor \mu/2 \rfloor} p'_{2i}(m) \mathsf{Pr}\left[n_0 = \mu - 2i\right] \overset{(53)}{\geq} -e^{-\mu/18} + \sum_{i=\lfloor \mu/4 \rfloor}^{\lfloor \mu/2 \rfloor} p'_{2i}(m) \mathsf{Pr}\left[n_0 = \mu - 2i \mid n_0 \leq \mu/2\right] \\
&\overset{(47)}{\geq} -e^{-\mu/18} + \sum_{i=\lfloor \mu/4 \rfloor}^{\lfloor \mu/2 \rfloor} \frac{\lambda_d}{\sqrt{2i}} \mathsf{Pr}\left[n_0 = \mu - 2i \mid n_0 \leq \mu/2\right] = \Theta\left(\frac{\lambda_d}{\sqrt{2\lfloor \mu/2 \rfloor}}\right) = \Theta(1/\sqrt{\mu}),
\end{aligned}$$

where d is a (asymptotic in μ) constant upper bound on $m/\sqrt{2\lfloor \mu/4 \rfloor} = 2\lceil c\sqrt{\mu}/2 \rceil / \sqrt{2\lfloor \mu/4 \rfloor}$, so that we can use (47).

Finally, (51) and (52) are immediate consequences of (50).

We establish a small piece of notation: for $a_1, \ldots, a_\mu \in \{-1, 0, 1\}$ and $X \subseteq [\mu]$, we abbreviate $\sum_{i \in X} a_i$ with $a(X)$. The following lemma is the "non-splitting" argument already mentioned in the informal proof of Theorem 5.

Lemma 19. *Let $\mu \in \mathbb{N}_{>0}$ and $a_1, \ldots, a_\mu \in \{-1, 0, 1\}$ be independently and uniformly distributed random variables. Let $\emptyset \subsetneq R \subsetneq S \subset [\mu]$. Let $c > 0$ and $t \in \mathbb{Z}$ with $|t| \leq c\sqrt{|S|}$ be arbitrary. Then*

$$\max_i \mathsf{Pr}\left[a(R) = i \mid a(S) = t\right] \leq \frac{1}{1 + \frac{\lambda_1 \lambda_{c+1}}{\Lambda^2}}. \tag{54}$$

Proof. Without loss of generality, assume $t \geq 0$; the case $t < 0$ is symmetric. Fix a value i^* for i that maximizes the probability in (54). We first claim that we can assume $0 \leq i^* \leq t$ without loss of generality. To see this, consider

$$\begin{aligned}\mathsf{Pr}\left[a(R) = i^* \mid a(S) = t\right] &= \frac{\mathsf{Pr}\left[a(R) = i^* \wedge a(S) = t\right]}{\mathsf{Pr}\left[a(S) = t\right]} \\ &= \frac{\mathsf{Pr}\left[a(R) = i^* \wedge a(S \setminus R) = t - i^*\right]}{\mathsf{Pr}\left[a(S) = t\right]} = \frac{\mathsf{Pr}\left[a(R) = i^*\right] \mathsf{Pr}\left[a(S \setminus R) = t - i^*\right]}{\mathsf{Pr}\left[a(S) = t\right]}. \end{aligned} \tag{55}$$

If $i^* < 0$, then $\mathsf{Pr}\left[a(R) = i^*\right] \leq \mathsf{Pr}\left[a(R) = 0\right]$ and $\mathsf{Pr}\left[a(S \setminus R) = t - i^*\right] \leq \mathsf{Pr}\left[a(S \setminus R) = t - 0\right]$ by (49), so we can set $i^* = 0$ as a value that maximizes (55). Similarly, $i^* > t$ implied $\mathsf{Pr}\left[a(R) = i^*\right] \leq \mathsf{Pr}\left[a(R) = t\right]$ and $\mathsf{Pr}\left[a(S \setminus R) = t - i^*\right] \leq \mathsf{Pr}\left[a(S \setminus R) = t - t\right]$, so we can use $i^* = t$ instead. Hence, we can assume $0 \leq i^* \leq t$. In fact, we may assume that:
(a) $i^* \leq c\sqrt{|R|}$, or
(b) $t - i^* \leq c\sqrt{|S \setminus R|}$.
Namely, if neither (a) nor (b) were satisfied, we would have the contradiction

$$t = i^* + (t - i^*) > c\sqrt{|R|} + c\sqrt{|S \setminus R|} > c\sqrt{|R| + |S \setminus R|} = c\sqrt{|S|} \geq t.$$

If (a) holds, then $i^* + 1 \leq c\sqrt{|R|} + 1 \leq (c+1)\sqrt{|R|}$, so

$$\mathsf{Pr}\left[a(R) = i^* + 1\right] = p_{|R|}(i^* + 1) \overset{(51)}{\geq} \frac{\lambda_{c+1}}{\Lambda} p_{|R|}(i^*) = \frac{\lambda_{c+1}}{\Lambda} \mathsf{Pr}\left[a(R) = i^*\right]. \tag{56}$$

Furthermore,

$$\mathsf{Pr}\left[a(S \setminus R) = t - (i^* + 1)\right] = p_{|S \setminus R|}(t - (i^* + 1)) \geq \frac{\lambda_1}{\Lambda} p_{|S \setminus R|}(t - i^*) = \frac{\lambda_1}{\Lambda} \mathsf{Pr}\left[a(s \setminus R) = t - i^*\right] \tag{57}$$

either trivially by (49) (in case $i^* < t$, and using that $\lambda_1 \leq \Lambda$), or by (51) (in case $i^* = t$, and using that then $|t - i^*|, |t - (i^* + 1)| \leq 1 \leq \sqrt{|S \setminus R|}$). Combining (56,57) yields

$$\mathsf{Pr}\left[a(R) = i^* + 1 \wedge a(S) = t\right] \geq \frac{\lambda_1 \lambda_{c+1}}{\Lambda^2} \mathsf{Pr}\left[a(R) = i^* \wedge a(S) = t\right],$$

whence

$$\begin{aligned}\max_i \mathsf{Pr}\left[a(R) = i \mid a(S) = t\right] &= \frac{\mathsf{Pr}\left[a(S) = t \wedge a(R) = i^*\right]}{\mathsf{Pr}\left[a(S) = t\right]} \leq \frac{\mathsf{Pr}\left[a(S) = t \wedge a(R) = i^*\right]}{\mathsf{Pr}\left[a(R) \in \{i^*, i^* + 1\} \wedge a(S) = t\right]} \\ &\leq \frac{\mathsf{Pr}\left[a(S) = t \wedge a(R) = i^*\right]}{\mathsf{Pr}\left[a(R) = i^* \wedge a(S) = t\right] + \mathsf{Pr}\left[a(R) = i^* + 1 \wedge a(S) = t\right]} \leq \frac{1}{1 + \frac{\lambda_1 \lambda_{c+1}}{\Lambda^2}},\end{aligned}$$

which shows (54). The case (b) is symmetric.

Lemma 20. *Let $\mu \in \mathbb{N}_{>0}$ and $a_1, \ldots, a_\mu \in \{-1, 0, 1\}$ be independently and uniformly distributed random variables. Assume fixed nonempty sets $X, Y \subseteq [\mu]$. Define $\Delta_X := X \setminus Y$, $\Delta_Y := Y \setminus X$, and $\Delta_{XY} = X \cap Y$. Denote by* SMALL *the event that*

$$a(\Delta_X), a(\Delta_Y), a(\Delta_{XY}) \leq \sqrt{\min\{|\Delta_X|, |\Delta_Y|, |\Delta_{XY}|\}} + 1$$

Then

$$\Pr[a(X) = a(Y) = 1 \wedge \text{SMALL}] \geq \frac{2\lambda_1\lambda_2\Gamma_1}{\mu} \tag{58}$$

Proof. Note that $\Delta_X \cup \Delta_Y \cup \Delta_{XY} = X \cup Y$, where the union on the left-hand side is disjoint. First, we treat the case $|\Delta_{XY}| \geq |\Delta_X|, |\Delta_Y|$. In this case, we assume without loss of generality $|\Delta_X| \geq |\Delta_Y|$ and hence $|\Delta_{XY}| \geq |\Delta_X| \geq |\Delta_Y|$. Let E denote the event that $|a(\Delta_Y)| \leq \sqrt{|\Delta_Y|}$, and let F denote the event that $a(\Delta_X) = a(\Delta_Y)$. We obtain

$$\Pr[E] = \Pr\left[\left|\sum_{j \in \Delta_Y} a_j\right| \leq \sqrt{|\Delta_Y|}\right] \overset{(52)}{\geq} \Gamma_1 \tag{59}$$

and

$$\Pr[F \mid E] \geq \min_{|i| \leq |\Delta_Y|} \Pr[a(\Delta_X) = i \mid E] \overset{(*)}{=} \min_{|i| \leq |\Delta_Y|} \Pr[a(\Delta_X) = i]$$

$$= \min_{|i| \leq |\Delta_Y|} p_{|\Delta_X|}(i) \underset{|\Delta_Y| \leq |\Delta_X|}{\overset{(50)}{\geq}} \frac{\lambda_1}{\sqrt{|\Delta_X|}}, \tag{60}$$

where $(*)$ uses that E is independent of $a(\Delta_X)$. Combining (59,60) gives

$$\Pr[E \wedge F] = \Pr[F \mid E] \Pr[E] \geq \lambda_1\Gamma_1 / \sqrt{|\Delta_X|}. \tag{61}$$

Now since $E \wedge F$ implies $a(X) = a(Y)$ as well as $|a(\Delta_X)| = |a(\Delta_Y)| \leq \sqrt{|\Delta_Y|} \leq \sqrt{|\Delta_{XY}|}$,

$$\Pr[a(X) = a(Y) = 1 \mid E \wedge F] = \Pr[a(\Delta_{XY}) = 1 - a(\Delta_X) \mid E \wedge F]$$

$$\geq \min_{|i| \leq \sqrt{|\Delta_Y|}+1} \Pr[a(\Delta_{XY}) = i \mid E \wedge F] \overset{(*)}{=} \min_{|i| \leq \sqrt{|\Delta_Y|}+1} \Pr[a(\Delta_{XY}) = i]$$

$$= \min_{|i| \leq \sqrt{|\Delta_Y|}+1} p_{|\Delta_{XY}|}(i) \underset{\sqrt{|\Delta_Y|}+1 \leq 2\sqrt{|\Delta_{XY}|}}{\overset{(50)}{\geq}} \frac{\lambda_2}{\sqrt{|\Delta_{XY}|}}, \tag{62}$$

where $(*)$ uses that $E \wedge F$ is independent of $a(\Delta_{XY})$. Now observe that $a(X) = a(Y) = 1 \wedge E \wedge F$ implies $|a(\Delta_X)| = |a(\Delta_Y)| \leq \sqrt{|\Delta_Y|}$ along with $|a(\Delta_{XY})| = |1 - a(\Delta_Y)| \leq \sqrt{|\Delta_Y|} + 1$. Hence, $a(X) = a(Y) = 1 \wedge E \wedge F$ implies SMALL, and we obtain

$$\Pr[a(X) = a(Y) = 1 \wedge \text{SMALL}] \geq \Pr[a(X) = a(Y) = 1 \wedge E \wedge F]$$

$$= \Pr[a(X) = a(Y) = 1 \mid E \wedge F] \Pr[E \wedge F] \overset{(61,62)}{\geq} \frac{\lambda_1\lambda_2\Gamma_1}{\sqrt{|\Delta_X| \cdot |\Delta_{XY}|}} \overset{(*)}{\geq} \frac{2\lambda_1\lambda_2\Gamma_1}{\mu}$$

as desired, where $(*)$ uses that $|\Delta_X| + |\Delta_{XY}| = |X| \leq \mu$ and hence[12] $|\Delta_X| \cdot |\Delta_{XY}| \leq (\mu/2)^2$.

The cases $|\Delta_X| \geq |\Delta_{XY}|, |\Delta_Y|$ and $|\Delta_Y| \geq |\Delta_{XY}|, |\Delta_X|$ can be treated analogously.

[12] for $a, b \in \mathbb{R}$, we have $a^2 - 2ab + b^2 = (a-b)^2 \geq 0 \Rightarrow a^2 + 2ab + b^2 = (a+b)^2 \geq 4ab \Rightarrow ((a+b)/2)^2 \geq ab$

Lemma 21. *In the situation of Lemma 20, let additionally $Z \subseteq [\mu]$, $Z \neq \emptyset, X, Y$. Then*

$$\mathsf{Pr}\left[a(Z) \neq 1 \mid a(X) = a(Y) = 1 \land \text{SMALL}\right] \geq \frac{\lambda_1\lambda_2}{\lambda_1\lambda_2 + \Lambda^2}. \tag{63}$$

Proof. Let $Z_X := Z \cap \Delta_X$, $Z_Y := Z \cap \Delta_Y$, and $Z_{XY} := Z \cap \Delta_{XY}$. Write G shorthand for the event $a(X) = a(Y) = 1 \land \text{SMALL}$.

Now first, if $Z \neq Z_X \cup Z_Y \cup Z_{XY}$, then there is an index $j \in Z \setminus (X \cup Y)$, and hence

$$\mathsf{Pr}\left[a(Z) \neq 1 \mid G\right] = \mathsf{Pr}\left[a_j \neq 1 - a(Z \setminus \{j\}) \mid G\right] \geq \min_{|i| \leq 1} \mathsf{Pr}\left[a_j \neq i \mid G\right] \overset{(*)}{=} \min_{|i|\leq 1} \mathsf{Pr}\left[a_j \neq i\right] = 2/3.$$

Here, $(*)$ uses the fact that G and a_j are independent. Since $0 < \lambda_c \leq \Lambda$ for all c, we have $2/3 \geq 1/2 \geq \frac{\lambda_1\lambda_2}{\lambda_1\lambda_2+\Lambda^2}$, and (63) follows. Hence, we may assume that Z completely decomposes into Z_X, Z_Y, and Z_{XY}.

Next, assume $Z_X \neq \emptyset, \Delta_X$, so $\emptyset \subsetneq Z_X \subsetneq \Delta_X$. Observe that for mutually exclusive events B_i with $\mathsf{Pr}\left[\bigvee_i B_i\right] = 1$, and arbitrary A, we have

$$\mathsf{Pr}\left[A\right] = \sum_i \mathsf{Pr}\left[A \land B_i\right] = \sum_i \mathsf{Pr}\left[A \mid B_i\right] \mathsf{Pr}\left[B_i\right] \leq \max_i \mathsf{Pr}\left[A \mid B_i\right] \sum_i \mathsf{Pr}\left[B_i\right] = \max_i \mathsf{Pr}\left[A \mid B_i\right]. \tag{64}$$

Since G implies $|a(\Delta_X)| \leq \sqrt{|\Delta_X|}$, we obtain

$$\begin{aligned}
\mathsf{Pr}\left[a(Z) = 1 \mid G\right] &\overset{(64)}{\leq} \max_{|t| \leq \sqrt{|\Delta_X|}} \mathsf{Pr}\left[a(Z) = 1 \mid G \land a(\Delta_X) = t\right] \\
&\overset{(*)}{=} \max_{\substack{|t| \leq \sqrt{|\Delta_X|} \\ i}} \mathsf{Pr}\left[a(Z_X) = i \mid G \land a(\Delta_X) = t\right] \overset{(*)}{=} \max_{\substack{|t| \leq \sqrt{|\Delta_X|} \\ i}} \mathsf{Pr}\left[a(Z_X) = i \mid a(\Delta_X) = t\right] \overset{(\dagger)}{\leq} \frac{1}{1 + \frac{\lambda_1\lambda_2}{\Lambda^2}}
\end{aligned}$$

which implies (63). Here, $(*)$ uses that G only depends on $a(\Delta_X)$ (but not on the individual a_j for $j \in \Delta_X$), and $(\dagger)$ uses Lemma 19 with $R = Z_X$, $S = \Delta_X$. Analogous reasoning shows that this holds also when $Z_Y \neq \emptyset, \Delta_Y$ and when $Z_{XY} \neq \emptyset, \Delta_{XY}$.

So far we have shown (63) unless all of the following conditions are fulfilled: $Z = Z_X \cup Z_Y \cup Z_{XY}$, $Z_X \in \{\emptyset, \Delta_X\}$, $Z_Y \in \{\emptyset, \Delta_Y\}$, and $Z_{XY} \in \{\emptyset, \Delta_{XY}\}$. That leaves only the following remaining possibilities:

- $Z = X$, or $Z = Y$, or $Z = \emptyset$: this cannot happen by assumption.
- $Z = \Delta_X$ or $Z = \Delta_Y$ or $Z = \Delta_{XY}$: using Lemma 19 (e.g., in case $Z = \Delta_X$ with $R = Z = \Delta_X$ and $S = X$) shows (63).
- $Z = \Delta_X \cup \Delta_Y \cup \Delta_{XY} = X \cup Y$: we can use Lemma 19 with $R = X$ and $S = Z = X \cup Y$ to show (63).
- $Z = \Delta_X \cup \Delta_Y$: in this case, $a(X) = a(Y) = 1$ would imply $a(\Delta_X) + a(\Delta_{XY}) = a(X) = a(Y) = a(\Delta_Y) + a(\Delta_{XY})$, whence $a(\Delta_X) = a(\Delta_Y)$. Thus $a(Z) = a(\Delta_X) + a(\Delta_Y) = 2a(\Delta_X) \neq 1$ always, and Lemma 19 follows.

Summarizing, this shows (63) in general.

Now we can combine Lemma 20 and Lemma 21 to obtain

Theorem 22. *Let $\mu \in \mathbb{N}_{>0}$ and $a_1, \ldots, a_\mu \in \{-1, 0, 1\}$ be independently and uniformly distributed random variables. Assume fixed nonempty sets $X, Y, Z \subseteq [\mu]$ with $Z \neq X, Y$. Then*

$$\mathsf{Pr}\left[a(X) = a(Y) = 1 \neq a(Z)\right] \geq \frac{\lambda_1^2\lambda_2^2\Gamma_1}{\lambda_1\lambda_2 + \Lambda^2} \cdot \frac{1}{\mu}.$$

This finally proves Theorem 5 if we just adapt notation: in the situation of the proof sketch of Theorem 5 and Definition 1, set $X = X_1$, $Y = X_2$, $Z = Z_1$, and $\mu = \ell$, then apply Theorem 22. (Note that at this point, we crucially use that we have hardwired $a_0 := -1$, so that, e.g., $a_{X_1} = a(X) - 1$, and thus $a_{X_1} = 0 \Leftrightarrow a(X) = 1$.)

A.2 Proof of Theorem 6

Proof. We use $\mathsf{PHF.Gen}$ and $\mathsf{PHF.TrapGen}$ algorithms similar to those from Theorem 5. First, let $J = J(k)$ be a positive function (we will optimize the choice of J later). Then define

- $\mathsf{PHF.TrapGen}(1^k, g, h)$ chooses uniformly and independently $a_{ij} \in \{-1, 0, 1\}$ for $1 \leq i \leq \ell$ and $1 \leq j \leq J$, as well as random group exponents $b_0, \ldots, b_\ell$. It sets $a_i = \sum_{j=1}^{J} a_{ij}$ and then $h_0 = g^{-1}h^{b_0}$ and $h_i = g^{a_i}h^{b_i}$ for all i. It finally returns $K = (h_0, \ldots, h_\ell)$ and $t = (b_0, a_1, b_1, \ldots, a_\ell, b_\ell)$.
- $\mathsf{PHF.TrapEval}(t, X)$ parses $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ and returns $a = -1 + \sum_{i=1}^{\ell} a_i x_i$ and $b = b_0 + \sum_{i=1}^{\ell} b_i x_i$.

The main difference to the functions from Theorem 5 is that the a_i are not chosen from $\{-1, 0, 1\}$ but instead in turn as random walks of length J. Now adding r independent random walks of length J just yields a random walk of length rJ. Hence, we obtain that for all keys K', all $X \in \{0,1\}^\ell$, and for the exponent a_X output by $\mathsf{PHF.TrapEval}(t, X)$:

$$\Theta\left(1/\sqrt{\ell J}\right) \leq \Pr\left[a_X = 0\right] \leq \Theta\left(1/\sqrt{J}\right),$$

and with techniques from Appendix A.1, we obtain for all $X, Y \in \{0,1\}^\ell$ with $X \neq Y$:

$$\Pr\left[a_Z = 0 \mid a_X = 0\right] = \Theta(1/\sqrt{J})$$

Hence for all $X_1, Z_1, \ldots, Z_q$, we have

$$\begin{aligned}\Pr\left[a_{X_1} = 0 \wedge a_{Z_1}, \ldots, a_{Z_q} \neq 0\right] &= \Pr\left[a_{X_1} = 0\right] \Pr\left[a_{Z_1}, \ldots, a_{Z_q} \neq 0 \mid a_{X_1} = 0\right] \\ &\geq \Theta(1/\sqrt{\ell J}) \left(1 - \sum_{i=1}^{q} \Pr\left[a_{Z_i} = 0 \mid a_{X_1} = 0\right]\right) \geq \Theta(1/\sqrt{\ell J})(1 - q\Theta(1/\sqrt{J})).\end{aligned}$$

Setting J suitably in the order of q^2 proves the theorem.

A.3 Proof of Theorem 7

Proof. Fix PPT algorithms $\mathsf{PHF.TrapGen}$ and $\mathsf{PHF.TrapEval}$ and assume $\ell = 2$ without loss of generality. Consider $X_1 = (1,1)$, $X_2 = (1,0)$, $X_3 = (0,0)$, and $Z_1 = (0,1)$. Assume that K', t have been generated via $\mathsf{PHF.TrapGen}(1^k, g, h)$ for uniform $g, h \in \mathbb{G}$. Define (a_X, b_X) for $X \in \{0,1\}^\ell$ as the result of $\mathsf{PHF.TrapEval}(t, X)$. Assume that $a_{X_1} = a_{X_2} = a_{X_3} = 0$, which implies that

$$\mathrm{H}^{\mathsf{MG}}_{K'}(X_1) = h_0 h_1 h_2 = h^{b_{X_1}},\ \mathrm{H}^{\mathsf{MG}}_{K'}(X_2) = h_0 h_1 = h^{b_{X_2}},\ \mathrm{H}^{\mathsf{MG}}_{K'}(X_3) = h_0 = h^{b_{X_3}}.$$

We will show now that $a_{Z_1} \neq 0$ allows to efficiently compute $\mathrm{dlog}_h(g)$, which proves the theorem. Namely, $a_{Z_1} \neq 0$ implies

$$g^{a_{Z_1}} h^{b_{Z_1}} = \mathrm{H}^{\mathsf{MG}}_{K'}(Z_1) = h_0 h_2 = \frac{\mathrm{H}^{\mathsf{MG}}_{K'}(X_1) \cdot \mathrm{H}^{\mathsf{MG}}_{K'}(X_3)}{\mathrm{H}^{\mathsf{MG}}_{K'}(X_2)}.$$

Considering the discrete logarithms to base h yields

$$\mathrm{dlog}_h(g) a_{Z_1} + b_{Z_1} = b_{X_1} - b_{X_2} + b_{X_3} \mod |G|$$

and hence, whenever $a_{Z_1} \neq 0$ and $|G|$ is known and prime, we can efficiently obtain $\mathrm{dlog}_h(g)$, solving the discrete logarithm problem for h and g.

B Randomized Programmable Hash Functions

B.1 Definitions

A *randomized group hash function* RH $=$ (RPHF.Gen, RPHF.Eval) for a group family $G = (\mathbb{G}_k)$ and with input length $\ell = \ell(k)$ and randomness space $\mathcal{R} = (\mathcal{R}_k)$ consists of two PPT algorithms. For security parameter $k \in \mathbb{N}$, a key $K \stackrel{\$}{\leftarrow} \mathsf{RPHF.Gen}(1^k)$ is generated by the key generation algorithm RPHF.Gen. This key K can then be used for the deterministic evaluation algorithm RPHF.Eval to evaluate RH via $y \leftarrow \mathsf{RPHF.Eval}(K, X; r) \in \mathbb{G}$ for any $X \in \{0,1\}^\ell$ and $r \in \mathcal{R}$. We write $\mathrm{RH}_K(X; r) = \mathsf{RPHF.Eval}(K, X; r)$.

Definition 23. *A randomized group hash function* RH *is an* (m, n, γ, δ)-programmable randomized hash function *if there are PPT algorithms* RPHF.TrapGen *(the trapdoor key generation algorithm),* RPHF.TrapEval *(the deterministic trapdoor evaluation algorithm), and* RPHF.TrapRand *(the deterministic randomness generator) such that the following holds:*

Syntactics: *For $g, h \in \mathbb{G}$, the trapdoor key generation $(K', t) \stackrel{\$}{\leftarrow} \mathsf{RPHF.TrapGen}(1^k, g, h)$ outputs a key K' and a trapdoor t. Trapdoor evaluation $(a(\cdot), b(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(t, X)$ produces two deterministic polynomial-time functions $a(\cdot)$ and $b(\cdot)$, for any $X \in \{0,1\}^\ell$. Moreover, $r \leftarrow \mathsf{RPHF.TrapRand}(t, X, i)$ produces an element r from $\mathcal{R}$, for any $X \in \{0,1\}^\ell$ and index $1 \leq i \leq m$.*

Correctness: *We demand $\mathrm{RH}_{K'}(X; r) = \mathsf{RPHF.Eval}(K', X; r) = g^{a(r)} h^{b(r)}$ for all $g, h \in \mathbb{G}$ and all possible $(K', t) \stackrel{\$}{\leftarrow} \mathsf{RPHF.TrapGen}(1^k, g, h)$, for all $X \in \{0,1\}^\ell$ and $1 \leq i \leq m$, $(a(\cdot), b(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(t, X)$, and for $r \leftarrow \mathsf{RPHF.TrapEval}(t, X, i)$.*

Statistically close trapdoor keys: *For $K \stackrel{\$}{\leftarrow} \mathsf{RPHF.Eval}(1^k)$ and $(K', t) \stackrel{\$}{\leftarrow} \mathsf{RPHF.Eval}(1^k)$, the keys K and K' are statistically γ-close: $K \stackrel{\gamma}{\equiv} K'$.*

Close to uniform randomness: *For all $g, h \in \mathbb{G}$ and all K' in the range of (the first component of) $\mathsf{RPHF.TrapGen}(1^k, g, h)$, for all $X_1, \ldots, X_m$, and $r_{X_i} \leftarrow \mathsf{RPHF.TrapRand}(t, X_i, i)$, the r_{X_i} are distributed statistically γ-close to independently uniform over $\mathcal{R}$ (over all possible t).*

Well-distributed logarithms: *For all $g, h \in \mathbb{G}$ and all K' in the range of (the first component of) $\mathsf{RPHF.TrapGen}(1^k, g, h)$, for all $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^\ell$ with $X_i \neq Z_j$ for any i, j, for all $\tilde{r}_1, \ldots, \tilde{r}_n \in \mathcal{R}$, and for all $(a_{X_i}(\cdot), b_{X_i}(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(t, X_i)$, $r_{X_i} \leftarrow \mathsf{RPHF.TrapRand}(t, X_i, i)$ and $(a_{Z_i}(\cdot), b_{Z_i}(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(t, Z_i)$, we have*

$$\Pr\left[a_{X_1}(r_{X_1}) = \ldots = a_{X_m}(r_{X_m}) = 0 \quad \wedge \quad a_{Z_1}(\tilde{r}_1), \ldots, a_{Z_n}(\tilde{r}_n) \neq 0\right] \geq \delta, \tag{65}$$

where the probability is over the trapdoor t that was produced along with K'. Here X_i may depend on all X_j and r_{X_j} for $j < i$, and the $Z_1, \ldots, Z_n$ may depend on all X_i and r_{X_i}.

If γ is negligible and δ is noticeable, we simply call RH *(m, n)-programmable.*

We remark that RPHFs are a strict generalization of PHFs from Section 3. Furthermore, it can be verified that our two applications of PHFs from Section 4 can also be securely instantiated with RPHFs.

B.2 Construction

In the following we denote $[x]_{2^\ell} := x \bmod 2^\ell$. The first randomized programmable hash function is variant of a hash function implicitly used in a construction by Fischlin [33].

Definition 24. *Let $G = (\mathbb{G}_k)$ be a group family, and let $\ell = \ell(k)$ be a polynomial. Then we define the following group hash function $\mathrm{RH}^{\mathsf{F}} = (\mathsf{RPHF.Gen}, \mathsf{RPHF.Eval})$ with input length $\ell = \ell(k)$ and randomness space $\mathcal{R} = \{0,1\}^\ell$:*

- *$\mathsf{RPHF.Gen}(1^k)$ returns a uniformly and independently sampled $K = (h_0, h_1, h_2) \in \mathbb{G}^3$.*
- *$\mathsf{RPHF.Eval}(K, X; r)$ parses $K = (h_0, h_1, h_2) \in \mathbb{G}^3$, $X \in \{0,1\}^\ell$, $r \in \{0,1\}^\ell$, computes and returns*

$$\mathrm{RH}^{\mathsf{F}}_K(X; r) = h_0 h_1^r h_2^{[r+X]_{2^\ell}}$$

Theorem 25. *For any group $\mathbb{G}$ with known order,* RH^{F} *is a* $(1,1,0,1/2)$*-programmable randomized hash function.*

Proof. Consider the following algorithms:

– $\mathsf{RPHF.TrapGen}(1^k, g, h)$ chooses uniformly and independently $r_1 \in \{0,1\}^\ell$ and random group exponents b_0, b_1, b_2. It picks a random vector $\Delta = (\Delta_1, \Delta_2) \in \{(1,0),(0,1)\}$. It sets $h_0 = g^{-r_1}h^{b_0}$, $h_1 = g^{\Delta_1}h^{b_1}$, $h_2 = g^{\Delta_2}h^{b_2}$. It returns $K = (h_0, h_1, h_2)$ and $t = (r_1, b_0, b_1, b_2, \Delta)$.
– $\mathsf{RPHF.TrapEval}(t, X, 1)$: It defines and returns the functions $a(s)$ and $b(s)$ as $a(s) = -r_1 + \Delta_1 s + \Delta_2[s + X]_{2^\ell}$, $b(s) = b_0 + b_1 s + b_2[s + X]_{2^\ell}$.
– $\mathsf{RPHF.TrapRand}(t, X, 1)$: It computes and returns $r = \Delta_1 r_1 + \Delta_2[r_1 - X]_{2^\ell}$.

Clearly, $r_{X_1} \leftarrow \mathsf{RPHF.TrapRand}(t, X_1, 1)$ equals r_1 which is uniform random, for any K. We have to show that for all $X_1 \neq Z_1 \in \{0,1\}^\ell$, for all $\tilde{r}_1 \in \mathcal{R}$, and for the corresponding $(a_{X_1}(\cdot), b_{X_1}(\cdot), r_{X_1}) \leftarrow \mathsf{RPHF.TrapEval}(t, X_1, 1)$ and $(a_{Z_1}(\cdot), b_{Z_1}(\cdot)) \leftarrow \mathsf{RPHF.TrapEval}(t, Z_1, \perp)$, we have

$$\mathsf{Pr}\left[a_{X_1}(r_{X_1}) = 0 \quad \wedge \quad a_{Z_1}(\tilde{r}_1) \neq 0\right] \geq \delta.$$

By construction we have

$$a_{X_1}(r_{X_1}) = -r_1 + \Delta_1(\Delta_1 r_1 + \Delta_2[r_1 + X_1]_{2^\ell}) + \Delta_2(\Delta_1 r_1 + \Delta_2[[r_1 + X_1]_{2^\ell} - X_1]_{2^\ell}) = 0,$$

always, and independent of everything else. It leaves to consider $\Pr[a_{Z_1}(\tilde{r}_1) \neq 0]$. We distinguish between two cases. If $\tilde{r}_1 \neq r_{X_1}$, then

$$\Pr[a_{Z_1}(\tilde{r}_1) \neq 0] \geq \Pr[a_{Z_1}(\tilde{r}_1) \neq 0 \mid \Delta = (1,0)] \Pr[\Delta = (1,0)] = \frac{1}{2}\Pr[-r_1 + \tilde{r}_1 \neq 0] = \frac{1}{2},$$

since $\Delta = (1,0)$ implies $\tilde{r}_1 = r_{X_1} = r_1$. If $\tilde{r}_1 = r_{X_1}$, then

$$\begin{aligned}\Pr[a_{Z_1}(\tilde{r}_1) \neq 0] &\geq \Pr[a_{Z_1}(\tilde{r}_1) \neq 0 \mid \Delta = (0,1)] \Pr[\Delta = (0,1)] \\ &= \frac{1}{2}\Pr[-r_1 + [Z_1 + [r_1 - X_1]_{2^\ell}]_{2^\ell} \neq 0] = \frac{1}{2},\end{aligned}$$

since $\Delta = (0,1)$ implies $\tilde{r}_1 = r_{X_1} = [r_1 - X_1]_{2^\ell}$.

Again, the above theorem also generalizes to groups of unknown order.

Theorem 26. *For the group $\mathbb{G} = \mathrm{QR}_N$ of quadratic residues modulo $N = pq$ for safe distinct primes p and q, the function* RH^{F} *is a 2^ℓ-bounded $(1,1,3/N,1/2)$-programmable randomized hash function.*

Now if we just write things differently, we obtain the RPHF that was (implicitly) used in Okamoto's scheme from [57]. In particular, Okamoto's scheme can be explained as our bilinear signature scheme $\mathsf{SIG}_{\mathrm{BM}}[\mathrm{RH}]$, instantiated with a suitable RPHF RH^{L} over a cyclic group. Formally:

Definition 27. *Let $G = (\mathbb{G}_k)$ be a group family, where $\mathbb{G}_k$ is of order p_k. Then, we define* $\mathrm{RH}^{\mathsf{L}} = (\mathsf{RPHF.Gen}, \mathsf{RPHF.Eval})$ *as the following group hash function with randomness space $\mathcal{R} = \mathbb{Z}_{p(k)}$:*

– $\mathsf{RPHF.Gen}(1^k)$ *returns a uniformly and independently sampled $K = (h'_0, h'_1, h'_2) \in \mathbb{G}^3$.*
– $\mathsf{RPHF.Eval}(K, X; r)$ *parses $K = (h'_0, h'_1, h'_2) \in \mathbb{G}^3$, $X \in \mathbb{Z}_{p(k)}$, $r \in \mathcal{R}$, computes and returns*

$$\mathrm{RH}^{\mathsf{L}}_K(X; r) = h'_0 {h'_1}^r {h'_2}^X$$

The proof of the following theorem follows from the proof of Theorem 25 if we just set

$$h'_0 = h_0 \qquad\qquad h'_1 = h_1 \qquad\qquad h'_2 = h_2 h_1$$

and replace the computation modulo 2^ℓ in the exponent by a computation modulo the (known) group order $|\mathbb{G}|$.

Theorem 28. *For any group $\mathbb{G}$ with known order,* RH^{L} *is a* $(1, 1, 0, 1/2)$*-programmable randomized hash function.*

Again, the theorem also generalizes to groups $|\mathbb{G}|$ of unknown order where we have to statistically approximate the group order. In fact, we can even work with a significantly shorter randomness space:

Theorem 29. *For the group* $\mathbb{G} = \mathrm{QR}_N$ *of quadratic residues modulo* $N = pq$ *for safe distinct primes* p *and* q*, the function* RH^{L} *with randomness space* $\mathcal{R} = \{0, 1\}^L$ *for* $L \geq \ell + k$ *is a* 2^L*-bounded* $(1, 1, 3/N + 1/k, 1/2)$*-programmable randomized hash function.*

We can prove Theorem 29 using a trapdoor key setup similar to the one for the case of a known group order. Concretely, $\mathsf{RPHF.TrapGen}(1^k, g, h)$ tosses a random coin $\varDelta \in \{0, 1\}$ and sets up

$$h'_0 = g^{-r_1} h^{b_0} \qquad h'_1 = g h^{b_1} \qquad h'_2 = g^{\varDelta} h^{b_2}.$$

With this setup, we get in particular $a_X(r) = r + \varDelta X - r_1$. Hence, trapdoor randomness generation returns $r_1 - \varDelta X$. Because $r_1 \in \{0, 1\}^L$ for $L = \ell + k$ and $X \in \{0, 1\}^\ell$, this randomness value $r_1 - \varDelta X$ is statistically close to uniform even for $\varDelta = 1$.

In this way, we can explain the (implicit) RPHFs from the signature schemes of Camenisch and Lysyanskaya [21] (with $L = \ell + k + \log_2 N$) and a variant of Zhu [67] (with slightly larger randomness space $L = \ell + k$). Observe, however, that these constructions are not suitably bounded to achieve short signature schemes through Theorem 13. Recall that Theorem 13 assumed *bounded* RPHFs to ensure that certain exponents are coprime (so Lemma 14 can be used to extract a nontrivial root). In the schemes [21,67], a more direct investigation shows that even with the used (not suitably bounded) RPHFs, this coprimality holds with large probability.

We finally note that it is possible to generalize RH^{F}, resp. RH^{L} to an $(m, 1)$-RPHF. However, also this generalization is not sufficiently bounded in order to be useful to our applications of short signatures (cf. also Footnote 6).