

Programmable Hash Functions in the Multilinear Setting

Eduarda S.V. Freire^{1,*}, Dennis Hofheinz^{2,**},
Kenneth G. Paterson^{1,***}, and Christoph Striecks²

¹ Royal Holloway, University of London

² Karlsruhe Institute of Technology

Abstract. We adapt the concept of a programmable hash function (PHF, Crypto 2008) to a setting in which a multilinear map is available. This enables new PHFs with previously unachieved parameters.

To demonstrate their usefulness, we show how our (standard-model) PHFs can replace random oracles in several well-known cryptographic constructions. Namely, we obtain standard-model versions of the Boneh-Franklin identity-based encryption scheme, the Boneh-Lynn-Shacham signature scheme, and the Sakai-Ohgishi-Kasahara identity-based non-interactive key exchange (ID-NIKE) scheme. The ID-NIKE scheme is the first scheme of its kind in the standard model.

Our abstraction also allows to derive hierarchical versions of the above schemes in settings with multilinear maps. This in particular yields simple and efficient hierarchical generalizations of the BF, BLS, and SOK schemes. In the case of hierarchical ID-NIKE, ours is the first such scheme with full security, in either the random oracle model or the standard model.

While our constructions are formulated with respect to a generic multilinear map, we also outline the necessary adaptations required for the recent “noisy” multilinear map candidate due to Garg, Gentry, and Halevi.

Keywords: programmable hash functions, multilinear maps, identity-based encryption, identity-based non-interactive key exchange, digital signatures.

1 Introduction

Programmable Hash Functions. Programmable hash functions (PHFs) have been proposed in [18] as an abstraction of random oracles that can also be instantiated in the standard model. In a nutshell, a PHF H maps a bitstring X (e.g., a message to be signed) to a group element $H(X)$; a special trapdoor allows to decompose $H(X) = c^{aX} h^{bX}$ for previously chosen c, h . In a larger proof,

* Eduarda S.V. Freire was supported by CAPES Foundation/Brazil on grant 0560/09-0 and Royal Holloway, University of London.

** Dennis Hofheinz was supported by a DFG grant (GZ HO 4534/2-1).

*** Kenneth G. Paterson was supported by EPSRC Leadership Fellowship EP/H005455/1.

c will usually be a “challenge element” (e.g., a part of a given Diffie-Hellman challenge), so that $H(X)$ contains a challenge component if and only if $a_X \neq 0$.

PHFs can be used to employ partitioning strategies: e.g., Waters’ CDH-based signature scheme [24] (implicitly) uses a PHF to partition the set of all messages into “signable” and “unsignable” messages. (In his case, a message X is signable iff $a_X \neq 0$.) During the proof of unforgeability, we hope that all messages for which an adversary requests a signature are signable, while the adversary’s forgery corresponds to an unsignable message.

Limitations of PHFs. While initially meant as a standard-model replacement for random oracles, many applications require a degree of “programmability” that is not met by current PHF constructions. Technically, we have PHF constructions with $a_X \neq 0$ for most, but not all preimages X . Such PHFs are suitable, e.g., in certain signature or identity-based encryption schemes [24, 18].

However, several prominent schemes that are formulated in the random oracle model (e.g., [23, 4, 6]) would require a PHF with $a_X = 0$ for most (but not all) preimages. (Roughly speaking, in these schemes, adversarial queries X can be handled iff the corresponding hash does not have a challenge component, i.e., if $a_X = 0$.) Unfortunately, a recent result [17] shows that no black-box construction of such a PHF with $a_X = 0$ for most (but not all) X exists.

Our Work. We construct PHFs with $a_X = 0$ for most (but not all) X by slightly adapting the PHF definition to a setting in which a multilinear map is available.¹ We use our PHFs to give standard-model versions of prominent cryptographic schemes whose security has so far only been proven in the random oracle model. Specifically, we give standard-model versions of the Boneh-Franklin (BF) identity-based encryption scheme [4], Boneh-Lynn-Shacham (BLS) signatures [6], and the Sakai-Ohgishi-Kasahara (SOK) identity-based non-interactive key exchange (ID-NIKE) [23]. We also use our PHFs to realise a completely new secure cryptographic functionality: we present the first *fully secure* hierarchical ID-NIKE, with security either in the standard-model or the random oracle model. Our constructions assume the existence of an $\mathbf{O}(k)$ -linear map, where k is the security parameter.² We use an abstraction of multilinear maps that is compatible with the recent “noisy” candidate for multilinear maps of Garg, Gentry, and Halevi [13].

Some Technical Details. We circumvent the black-box impossibility result [17] by slightly adapting the PHF definition to a setting with multilinear maps. Intuitively, [17] uses that a_X is an exponent that can be viewed as a known function in certain unknown variables. This function is linear, because all involved group elements are from the same group, and only group operations are allowed. But the number of zeros of such a (nontrivial) linear function can be reasonably upper bounded. This contradicts the goal that $a_X = 0$ for many, but not all X .

¹ Concretely, we construct (poly, n)-MPHFs for any constant n . This denotes a slight variant of PHFs in a multilinear setting, with the following property. For any polynomial number of X_i and Z_1, \dots, Z_n (with $X_i \neq Z_j$), we have $a_{X_i} = 0$ and $a_{Z_j} \neq 0$ for all i, j with significant probability. The X_i, Z_j need not be known during setup.

² In fact, our optimizations only require a $\mathbf{O}(k/\log(k))$ -linear map.

By moving to a multilinear setting, we essentially allow (a limited number of) multiplications in the exponent. Hence, the exponent a_X is now no longer limited to be a linear function, but can be a *multivariate* polynomial. Such polynomials can have exponentially many zeros. For instance, we could choose secret values $\alpha_{i,b}$ (for $1 \leq i \leq |X|$ and $b \in \{0, 1\}$), such that exactly one element of each pair $(\alpha_{i,0}, \alpha_{i,1})$ is nonzero; say $\alpha_{i,b_i} \neq 0$. Then the function

$$a_X = \alpha(X) = \prod_{i=1}^{|X|} \alpha_{i,X_i} \quad (1)$$

(where X_i denotes the i -th bit of X) evaluates to zero everywhere except for $X = (b_1, \dots, b_{|X|})$. In fact, we implement a suitable variant of the function in (1) in the exponent (in the sense that $H(X) = c^{a_X} h^{b_X} = c^{\alpha(X)} h^{b_X}$ for a suitable blinding term h^{b_X}) through multilinear maps.³ In the process, we also recognize and refine an admissible hash function (AHF [3, 8, 1]) implicit in [19]. This yields the – by far – most efficient known AHFs. As a result, we get PHFs in the multilinear setting with $a_X = 0$ for many (but not all) X .

Applications. To demonstrate their power, we use our new PHFs to replace random oracles in three example applications. As one application, we obtain from BLS signatures [6] an existing standard-model signature scheme due to Boneh and Silverberg [5]; as a natural extension, we give a standard-model variant of the Boneh-Franklin IBE scheme [4]. However, our central application is the SOK [23] ID-NIKE scheme; from this scheme, we get the first fully secure ID-NIKE in the standard model.

In all cases, the analysis is completely modular: we prove the security of the PHF-based schemes solely from generic PHF properties. In particular, we can also view (programmable) random oracles as PHFs to obtain the original schemes, with essentially the original proofs.⁴ We view these results as strong evidence that PHFs are a useful abstraction of random oracles that also allows for standard-model instantiations.

In addition, we give natural *hierarchical* versions of all schemes in a setting with multilinear maps. (Recall that we require multilinear maps for our PHFs anyway.) Again, we can either use random oracles as PHFs to obtain reasonably efficient new schemes, or use our new PHFs to obtain (somewhat less efficient) standard-model versions.

More on Our ID-NIKE Schemes. In the signature and IBE applications, we mainly explain (and slightly improve) existing schemes through PHFs. While this already hints at the potential of our notion of PHFs, their actual usefulness

³ We stress that these ideas are not new; essentially the same function in the exponent has been considered by Boneh and Silverberg [5] for a concrete signature scheme, building on work of Lysyanskaya [19]. Our contribution here is an abstraction (along with a few quantitative optimizations) that enables new applications.

⁴ The exception is the SOK scheme, for which we only get a proof under a slightly stronger computational assumption.

in building novel cryptographic functionalities is best demonstrated by our application to ID-NIKE.

Loosely speaking, a non-interactive key exchange (NIKE) provides any two parties registered in the system with a unique shared key, *without any interaction*. For NIKE in the identity-based setting, there is a single master public key held by a trusted authority (TA); each party additionally gets an individual user secret key from the TA, and combines its secret key with the identity of the other party to compute the shared key. This primitive is a powerful one. For one thing, it implies secure IBE under a minor technical requirement [20]. More importantly, it has important applications in managing keys and enabling secure communications in mobile *ad hoc* and sensor networks, where the energy cost of communication is at a premium [14, 9]. In the hierarchical setting, H-ID-NIKE allows the same functionality, but also allows the TA's operations to be distributed over a hierarchy, which is well-suited to military and emergency response scenarios. The advantages of ID-NIKE, in terms of reducing communication costs and latency in a realistic adversarial environment, are demonstrated in [9]. For further discussion of applications of NIKE and ID-NIKE, see [14, 12].

However, ID-NIKE has proven surprisingly hard to instantiate in the standard model, even more so in a hierarchical setting. Currently, to the best of our knowledge, there is precisely one efficient, secure ID-NIKE scheme with a proof of security in the random oracle model, namely the SOK scheme [23] (with security models and analysis in [11, 20]). There are no schemes secure in the standard model. One might think that such schemes could easily be obtained from known standard-model-secure IBE schemes, but this is not the case; the essential technical barrier seems to be the randomised key derivation in these IBE schemes.

In the hierarchical setting, Gennaro *et al.* [14] constructed H-ID-NIKE schemes that are secure under certain classes of key exposure, but which do not offer *full security*, the desirable and natural generalisation of the existing ID-NIKE security notion from [20] to the hierarchical setting. Moreover, their schemes do not scale well to large numbers of levels. The same criticisms apply to earlier schemes [2, 21] on which the scheme of Gennaro *et al.* [14] is based. Indeed, one of the open problems left in [14] is to construct a H-ID-NIKE scheme with security against *not only* compromise of any number of leaves, *but also* against any number of nodes at higher levels of the hierarchy.⁵

By substituting the random oracles in the SOK scheme [23] with our new PHFs, we obtain the first secure ID-NIKE schemes in the standard model. Furthermore, our construction extends naturally to the hierarchical setting, yielding the first fully secure H-ID-NIKE schemes. The construction can be instantiated using random oracles to obtain a reasonably efficient scheme, or using PHFs for security in the standard model. In the full version, we also show how multilinear maps can be used to achieve security in the broader scenario of multiple TAs, and for shared keys among whole groups of parties.

⁵ We note that there are other papers claiming to solve this open problem (*eg.* [16]), but these can be easily shown to provide insecure schemes.

Note on the Recent Candidate for Multilinear Maps. Recently, Garg, Gentry, and Halevi [13] have announced a candidate for a family of cryptographically interesting multilinear maps. Their candidate is lattice-based, heavily relies on the notion of noise, and thus does not provide groups in the usual sense. We comment on the necessary adaptations of our schemes to their setting inside.

2 Preliminaries

Notation. For $n \in \mathbb{R}$, let $[n] := \{1, \dots, [n]\}$. Throughout the paper, $k \in \mathbb{N}$ denotes the security parameter. For a finite set \mathcal{S} , we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from \mathcal{S} . For sets $\mathcal{S}^1, \mathcal{S}^2, \dots$ and $n \in \mathbb{N}$, we write $\mathcal{S}^{\leq n} := \bigcup_{i=1}^n \mathcal{S}^i$. For a probabilistic algorithm A , we write $y \leftarrow A(x)$ for the process of running A on input x with uniformly chosen random coins, and assigning y the result. If A 's running time is polynomial in k , then A is called probabilistic polynomial-time (PPT). A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if it vanishes faster than the inverse of any polynomial (i.e., if $\forall c \exists k_0 \forall k \geq k_0 : |f(k)| \leq 1/k^c$). f is significant if it dominates the inverse of some polynomial (i.e., if $\exists c, k_0 \forall k \geq k_0 : f(k) \geq 1/k^c$).

Multilinear Maps. An ℓ -group system consists of ℓ cyclic groups $\mathbb{G}_1, \dots, \mathbb{G}_\ell$ of prime order p , along with bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \rightarrow \mathbb{G}_{i+j}$ for all $i, j \geq 1$ with $i + j \leq \ell$. Let g_i be a canonical generator of \mathbb{G}_i (included in the group's description). The map $e_{i,j}$ satisfies $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$ (for all $a, b \in \mathbb{Z}_p$). When i, j are clear, we will simply write e instead of $e_{i,j}$. It will also be convenient to abbreviate $e(h_1, \dots, h_j) := e(h_1, e(h_2, \dots, e(h_{j-1}, h_j) \dots))$ for $h_j \in \mathbb{G}_{i_j}$ and $i = (i_1 + i_2 + \dots + i_j) \leq \ell$. By induction, it is easy to see that this map is j -linear. Additionally, we define $e(g) := g$. Finally, it can also be useful to define the group $\mathbb{G}_0 = \mathbb{Z}_{|\mathbb{G}_1|}^+$ of exponents to which this pairing family naturally extends. In the following, we will assume an ℓ -group system $\mathcal{MPG}_\ell = \{\{\mathbb{G}_i\}_{i \in [\ell]}, p, \{e_{i,j}\}_{i,j \geq 1, i+j \leq \ell}\}$ generated by a *multilinear maps parameter generator* \mathcal{MG}_ℓ on input a security parameter 1^k .

The GGH Candidate. We currently do not have candidates for multilinear maps between groups with cryptographically hard problems. However, Garg, Gentry, and Halevi [13] (henceforth GGH) suggest a concrete candidate for an ‘‘approximation’’ of multilinear maps, named *graded encoding systems*. With the GGH candidate, group elements have a randomized (and thus non-unique) representation dubbed ‘‘encoding’’. While it is possible to extract a unique ‘‘canonical bitstring’’ from an encoding, it is not possible to perform further computations with this extracted bitstring. An encoding can be re-randomized (e.g., to hide the sequence of operations that were performed), but only at the cost of introducing an artificial ‘‘noise’’ term in the encoding. Further operations (and re-randomizations) on this group element cause the noise to grow; once this noise grows beyond a certain bound, encodings can no longer be worked with.⁶

⁶ We further ignore a (negligible) error probability in most of the GGH procedures. Technically, however, this leads to applications with, e.g., negligible correctness error.

Our Abstraction. For readability and universality, we will generally use the notation from the abstract notion of multilinear maps described above. When instantiated with the GGH candidate, operations are meant to occur on encodings, without implicit re-randomizations. In particular, e.g., g now denotes an encoding (not a group element). Additionally, we will employ the following notation to indicate necessary re-randomizations, extractions, and comparisons when using encodings instead of group elements.

- $g \leftarrow \mathbb{G}_i$ means choosing a random encoding g at level i . (This corresponds to uniformly choosing a group element from \mathbb{G}_i .) We assume that encodings g chosen in such a way have a low noise level, say, 1.
- $g \stackrel{\text{enc}}{=} h$ holds iff the encodings g and h match.
- $g \stackrel{\text{grp}}{=} h$ holds iff the *group elements* encoded by g and h match, that is, iff the GGH `isZero` procedure identifies $g^{-1}h$ as the neutral element.⁷
- $\text{reRand}_j(g)$ is the re-randomization of encoding g . This re-randomization increases the noise level to a certain, a-priori fixed bound j . For simplicity, and abstracting, we only consider noise levels $j \in \mathbb{N}$. If g 's noise level is already at least j (e.g., because g is the output of reRand_j), then randomization fails. We note that the distributions $\text{reRand}_j(g)$ and $\text{reRand}_j(h)$ are statistically close for any two encodings g, h with $g \stackrel{\text{grp}}{=} h$ and noise level less than j .
- $\text{ext}(g)$ denotes the canonical bitstring extracted from encoding g . We have $\text{ext}(g) = \text{ext}(h)$ for any g, h with $g \stackrel{\text{grp}}{=} h$ of sufficiently small noise level.

Like [13], we omit parameters (such as noise bounds) to computations; asymptotic parameters can be derived from the suggestions in [13, Section 4.2].

Hard Problems. The ℓ -MDDH assumption is: given $(g, g^{x_1}, \dots, g^{x_{\ell+1}})$, (for $g \leftarrow \mathbb{G}_1$ and uniform exponents x_i), the element $e(g^{x_1}, \dots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniform \mathbb{G}_ℓ -element. The $(\ell + 1)$ -power assumption is: given (g, g^x) (for $g \leftarrow \mathbb{G}_1$ and uniform x), the element $e(\underbrace{g^x, \dots, g^x}_{\ell \text{ times}})^x \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniformly chosen \mathbb{G}_ℓ -element.⁸

3 Programmable Hash Functions in the Multilinear Setting

3.1 Motivation

Programmable hash functions (PHFs) have been defined in [18] as a special type of a group hash function (i.e., a hash function with images in a group).

⁷ Technically, the GGH `isZero` procedure only allows to compare two encodings on the “highest level” ℓ . To compare two level- i encodings (for $i < \ell$), we can first “lift” both to level ℓ by pairing them with a nonzero level- $(\ell - i)$ element.

⁸ We note that in the GGH setting, all elements g^{x_i} (resp. g^x), and the challenge $e(g^{x_1}, \dots, g^{x_\ell})^{x_{\ell+1}}$ (resp. $e(g^x, \dots, g^x)^x$) are produced with knowledge of the exponents x, x_i as fully randomized but low-noise encodings.

Namely, the image $H(X)$ of a PHF can always be explained as $H(X) = c^{a_X} h^{b_X}$ for externally given c, h . Usually, c will be a “challenge element” (e.g., from a Diffie-Hellman-like problem), and h will be a “controlled element” (e.g., with known exponent relative to a fixed group generator) used for blinding purposes. Intuitively, we require that both the events $a_X = 0$ and $a_X \neq 0$ occur with significant probability. Even more, an (m, n) -PHF guarantees that with significant probability, $a_{X_i} = 0$ for *any* m given inputs X_i , while $a_{Z_j} \neq 0$ for *any* n given inputs Z_j (with $X_i \neq Z_j$ of course). This means that the $H(X_i)$ contain no challenge component, while all $H(Z_j)$ do.

For our purposes, we will strive to construct efficient (poly, n) -PHFs for constant n (i.e., group hash functions which are $(q(k), n)$ -PHFs for any polynomial q). However, there are indications that such PHFs do not exist [17], at least according to the original definition from [18]. Thus, we will adapt the definition of PHFs to the multilinear setting, and construct the “multilinear analog” of a (poly, n) -PHF. Concretely, an (m, n) -PHF maps to a “target” group \mathbb{G}_ℓ . Here instead of explaining $H(X)$ as a product $c^{a_X} h^{b_X}$ for c, h in the target group \mathbb{G}_ℓ (as the case of PHFs), we will explain $H(X)$ as a product $e(c_1, \dots, c_\ell)^{a_X} e(B_X, h)$, for externally given challenges $c_i \in \mathbb{G}_1$ (which means $c = e(c_1, \dots, c_\ell) \in \mathbb{G}_\ell$) and controlled $h \in \mathbb{G}_1$. Note that the coefficient b_X in the usual definition of a PHF now becomes a preimage $B_X \in \mathbb{G}_{\ell-1}$ under a pairing operation.

3.2 Definitions

Definition 1 (Group hash function). A group hash function H into \mathbb{G} consists of two polynomial-time algorithms: the probabilistic algorithm $H\text{Gen}(1^k)$ outputs a key hk , and $H\text{Eval}(hk, X)$ (for a key hk and $X \in \{0, 1\}^k$) deterministically outputs an image $H_{hk}(X) \in \mathbb{G}$.

Definition 2 (MPHF). Assume an ℓ' -group system $\text{MPG}_{\ell'}$ as generated by $\text{MG}_{\ell'}(1^k)$. Let H be a group hash function into \mathbb{G}_ℓ ($\ell \leq \ell'$), and let $m, n \in \mathbb{N}$. We say that H is an (m, n) -programmable hash function in the multilinear setting ((m, n) -MPHF) if there are PPT algorithms $T\text{Gen}$ and $T\text{Eval}$ as follows.

- $T\text{Gen}(1^k, c_1, \dots, c_\ell, h)$ (for $c_i, h \in \mathbb{G}_1$ and $h \stackrel{\text{grp}}{\neq} 1$) outputs a key hk and a trapdoor td . We require that for all c_i, h , the distribution of hk is statistically close to the output of $H\text{Gen}$.⁹
- $T\text{Eval}(td, X)$ (for a trapdoor td and $X \in \{0, 1\}^k$) deterministically outputs $a_X \in \mathbb{Z}$ and $B_X \in \mathbb{G}_{\ell-1}$ with $H_{hk}(X) \stackrel{\text{grp}}{=} e(c_1, \dots, c_\ell)^{a_X} \cdot e(B_X, h)$. We require that there is a polynomial $p(k)$ such that for all hk and $X_1, \dots, X_m, Z_1, \dots, Z_n \in \{0, 1\}^k$ with $\{X_i\}_i \cap \{Z_j\}_j = \emptyset$,

$$P_{hk, \{X_i\}, \{Z_j\}} := \Pr [a_{X_1} = \dots = a_{X_m} = 0 \wedge a_{Z_1}, \dots, a_{Z_n} \neq 0] \geq 1/p(k), \quad (2)$$

⁹ There is a subtlety here: in case of encoded group elements, the output of $T\text{Gen}$ may consist of group elements whose noise level depends on the noise level of the c_i or h . Hence, we will assume a known a-priori bound on the noise level of the c_i and h . This assumption will be fulfilled in our applications.

where the probability is over possible trapdoors td output by TGen along with the given hk . Furthermore, we require that $P_{hk, \{X_i\}, \{Z_j\}}$ is close to statistically independent of hk . (Formally, $|P_{hk, \{X_i\}, \{Z_j\}} - P_{hk', \{X_i\}, \{Z_j\}}| \leq \nu(k)$ for all hk, hk' in the range of TGen , all $\{X_i\}, \{Z_j\}$, and negligible $\nu(k)$.)

We say that H is a (poly, n) -MPHF if it is a $(q(k), n)$ -MPHF for every polynomial $q(k)$, analogously for (m, poly) -MPHFs.

Note that the TEval algorithm of an MPHF into \mathbb{G}_1 yields $B_X \in \mathbb{G}_0$, i.e., exponents B_X . In fact, in this case, the MPHF definition coincides with the original PHF definition from [18].

Readers interested only in how to use MPHFs in cryptographic constructions may safely skip the remainder of this section.

3.3 Warmup: Programmable Random Oracles as MPHFs

A *programmable* random oracle RO with images in \mathbb{G}_1 can be interpreted as a group hash function in the obvious way. (By “programmable”, we mean that during a security proof, we can freely and adaptively determine images of RO , even depending on the inputs of TGen . The only restriction of this programming is that images should appear uniformly and independently distributed to an adversary who sees only public information.) However, note for this modeling to make sense in the first place, we should require that we can hash into \mathbb{G}_1 .

Theorem 1 (PROs as (poly, n) -MPHFs). *A programmable random oracle RO (in the above sense) with images in \mathbb{G}_1 can be programmed to act as a (poly, n) -MPHF for any constant n .*

Proof (Proof sketch). Fix a polynomial $q = q(k)$. We show that RO is a (q, n) -MPHF (with empty hk). For each new preimage X , we program $\text{RO}(X) := c^{a_X} h^{B_X}$ for the inputs $c := c_1$ and h to TGen , and a uniformly chosen exponent $B_X \in \mathbb{G}_0 = \mathbb{Z}_{|\mathbb{G}_1|}$. We choose $a_X = 1$ with probability $1/2q$, and $a_X = 0$ otherwise. TEval outputs these a_X, B_X , assigning them as necessary for previously unqueried inputs X . For any pairwise different $X_1, \dots, X_q, Z_1, \dots, Z_n$, we thus have

$$\Pr [\forall i : a_{X_i} = 0 \wedge \forall j : a_{Z_j} \neq 0] = \left(1 - \frac{1}{2q}\right)^q \cdot \left(\frac{1}{2q}\right)^n \geq \frac{1}{2} \cdot \left(\frac{1}{2q}\right)^n,$$

which is significant for polynomial q and constant n . □

3.4 Ingredient: Efficient Admissible Hash Functions

At the heart of our standard-model constructions lies a primitive dubbed “admissible hash function” (AHF) [3]. Unfortunately, the AHFs from [3] are not very efficient (and in fact only achieve a weaker AHF definition, see [8]). However, luckily, an earlier work by Lysyanskaya [19] already contains an implicit and much more efficient AHF.

Intuitively, an AHF can be thought of as a combinatorial counterpart of (poly, 1)-(M)PHFs. An AHF input X is mapped to an image $\text{AHF}(X)$ in a way that X can fall in the set of controlled, CO , inputs (meaning that we know a trapdoor that allows to answer adversary’s queries for that input) or uncontrolled, UN , inputs (meaning that we do not know any trapdoor but hope to embed a challenge element). (Unlike with (M)PHFs, however, this is a purely combinatorial property.) An AHF guarantees that for any X_1, \dots, X_q, Z , with significant probability, all X_i are controlled, and Z is uncontrolled.

We now give a definition that is a somewhat simpler variant of the AHF definitions from [8, 1], and then show a result implicit in [19].

Definition 3 (AHF). For a function $\text{AHF} : \{0, 1\}^k \rightarrow R^\ell$ (with a finite set¹⁰ R and polynomial $\ell = \ell(k)$) and $K \in (R \cup \{\perp\})^\ell$, define the function $F_K : \{0, 1\}^k \rightarrow \{\text{CO}, \text{UN}\}$ through $F_K(X) = \text{UN} \iff \forall i : K_i = \text{AHF}(X)_i \vee K_i = \perp$, where $\text{AHF}(X)_i$ denotes the i -th component of $\text{AHF}(X)$.¹¹ We say that AHF is q -admissible if there exists a PPT algorithm KGen and a polynomial $p(k)$, such that for all $X_1, \dots, X_q, Z \in \{0, 1\}^k$ with $Z \notin \{X_i\}$,

$$\Pr[F_K(X_1) = \dots = F_K(X_q) = \text{CO} \wedge F_K(Z) = \text{UN}] \geq 1/p(k), \tag{3}$$

where the probability is over $K \leftarrow \text{KGen}(1^k)$. We say that AHF is an admissible hash function (AHF) if AHF is q -admissible for all polynomials $q = q(k)$.

Thus, X is controlled (i.e., $F_K(X) = \text{CO}$) if there is an i with $X_i \neq K_i \neq \perp$.

Theorem 2 ([19]). Assume a family of codes $\{C_k\}$ with $C_k : \{0, 1\}^k \rightarrow R^\ell$ denoting both the code and its encoding function. Suppose that C_k has minimum distance at least $c \cdot \ell$ for a fixed constant $c > 0$. (That is, $X_1 \neq X_2$ implies that the vectors $C_k(X_1)$ and $C_k(X_2)$ differ in $\geq c \cdot \ell$ positions.) Then $\{C_k\}$ is an AHF.

Proof. Let $q = q(k)$ be a polynomial. We need to devise a PPT algorithm KGen such that (3) holds. $\text{KGen}(1^k)$ sets $d := \lfloor (\ln 2q)/c \rfloor$ (so d is the smallest integer such that $(1 - c)^d \leq 1/2q$), and picks K uniformly among all elements from $(R \cup \{\perp\})^\ell$ with exactly d non- \perp components. Hence, the set $I := \{i \mid K_i \neq \perp\}$ is of size d .

Now fix $X_1, \dots, X_q, Z \in \{0, 1\}^k$ with $Z \notin \{X_i\}$. Our choice of K implies $\Pr[F_K(Z) = \text{UN}] = |R|^{-d}$. For any fixed i , we want to upper bound the probability $\Pr[F_K(X_i) = \text{UN} \mid F_K(Z) = \text{UN}]$. (This step loosely corresponds to [19, Lemma 4].) Hence, assume $F_K(Z) = \text{UN}$; note that this conditioning leaves the distribution of I uniform. Now $C_k(X_i)$ and $C_k(Z)$ differ in a set $\Delta \subseteq [\ell]$ of positions with $|\Delta| \geq c\ell$. Hence, $F_K(X_i) = \text{UN}$ is equivalent to $I \cap \Delta = \emptyset$. Thus,

$$\begin{aligned} \Pr[F_K(X_i) = \text{UN} \mid F_K(Z) = \text{UN}] &= \Pr[I \cap \Delta = \emptyset \mid F_K(Z) = \text{UN}] \\ &\leq (1 - c)^d \leq e^{-cd} \leq \frac{1}{2q}. \end{aligned}$$

¹⁰ One should have $R = \{0, 1\}$ in mind here. Larger R (e.g., $R = [k]$) lead to slightly less pairing-intensive constructions of MPHFs, see the paragraph before Theorem 4.

¹¹ That is, for $R = \{0, 1\}$, we have $F_K(X) = \text{CO}$ iff there is an i with $K_i = 1 - \text{AHF}(X)_i$.

A union bound over i gives $\Pr [\forall i : F_K(X_i) = \text{UN} \mid F_K(Z) = \text{UN}] \leq 1/2$, so that

$$\Pr [F_K(Z) = \text{UN} \wedge \forall i : F_K(X_i) = \text{CO}] \geq \frac{1}{2} \cdot |R|^{-d} \geq \frac{1}{2} \cdot \left(\frac{1}{2q}\right)^{\frac{1}{c \cdot \log |R|(e)}}$$

which is significant. □

3.5 Main Result: MPHFs from Multilinear Maps

Our main result in this section is a simple construction of a (poly, n)-MPHF from an AHF.

Construction 1 (MM). Let $\text{AHF} : \{0, 1\}^k \rightarrow R^\ell$ be an admissible hash function and assume an ℓ' -group system $\mathcal{MPG}_{\ell'}$. The group hash function MM into \mathbb{G}_ℓ ($\ell \leq \ell'$) is given by the following algorithms:

- $\text{HGen}(1^k)$ picks $\tilde{h}_{i,j} \leftarrow \mathbb{G}_1 \setminus \{1\}$ (for $(i, j) \in [\ell] \times R$), sets $h_{i,j} := \text{reRand}_2(\tilde{h}_{i,j})$, and outputs $hk := \{h_{i,j}\}_{i \in [\ell], j \in R}$.¹²
- $\text{HEval}(hk, X)$ computes $(t_1, \dots, t_\ell) := \text{AHF}(X)$ and outputs $\text{MM}_{hk}(X) := e(h_{1,t_1}, \dots, h_{\ell,t_\ell})$.

Theorem 3. *The group hash function MM above is a (poly, 1)-MPHF.*

Proof. Fix a polynomial $q = q(k)$. We need to exhibit TGen and TEval algorithms as in Definition 2. TGen($1^k, c_1, \dots, c_\ell, h$) invokes $K \leftarrow \text{KGen}(1^k)$ and, for all $(i, j) \in [\ell] \times R$ and uniform exponents $r_{i,j} \neq 0$, it sets up

$$h_{i,j} := \begin{cases} \text{reRand}_2(h^{r_{i,j}}) & \text{if } K_i \neq j \text{ and } K_i \neq \perp, \\ \text{reRand}_2(c_i^{r_{i,j}}) & \text{if } K_i = j \text{ or } K_i = \perp. \end{cases} \tag{4}$$

For now, assume $c_i \stackrel{\text{grp}}{\neq} 1$ for all i , so our setup yields a perfectly distributed key $hk := \{h_{i,j}\}_{i,j}$ that is in fact independent of K .¹³ The trapdoor is $td := ((c_i), h, K, (r_{i,j}))$.

TEval(td, X) computes $(t_1, \dots, t_\ell) := \text{AHF}(X)$ and distinguishes two cases:

Case $F_K(X) = \text{CO}$, i.e., there is at least an i^* with $K_{i^*} \neq t_{i^*}$ and $K_{i^*} \neq \perp$. If we set $a_X = 0$ and

$$B_X := e(h_{1,t_1}, \dots, h_{i^*-1,t_{i^*-1}}, h_{i^*+1,t_{i^*+1}}, \dots, h_{\ell,t_\ell})^{r_{i^*,t_{i^*}}},$$

for any chosen i^* , we can decompose $\text{MM}_{hk}(X) \stackrel{\text{grp}}{=} e(c_1, \dots, c_\ell)^{a_X} e(B_X, h)$.

¹² The additional re-randomization step guarantees that the noise levels in scheme and simulation are the same. The concrete noise level of re-randomized elements depends on the maximal noise considered in the arguments of TGen.

¹³ In case of randomized encodings, the distribution of hk in the simulation may (e.g., with the GGH candidate) only be statistically close to the one in the scheme.

Case $F_K(X) = \text{UN}$, i.e., $K_i = t_i$ or $K_i = \perp$ for all i . This means that $h_{i,t_i} \stackrel{\text{grp}}{=} c_i^{r_i,t_i}$ for all i , so $\text{MM}_{hk}(X) \stackrel{\text{grp}}{=} e(c_1, \dots, c_\ell)^{a_X} e(B_X, h)$ for $a_X = \prod_i r_{i,t_i}$ and $B_X := 1$.

The AHF property (3) implies (2). (Note that $P_{hk, \{X_i\}, \{Z\}}$ only depends on K but not on hk .)

Finally, in case $c_i \stackrel{\text{grp}}{=} 1$ for some i , we have $e(c_1, \dots, c_\ell) \stackrel{\text{grp}}{=} 1$. If we replace all c_i in (4) with h , we can explain any image $\text{MM}_{hk}(X) \stackrel{\text{grp}}{=} e(h, \dots, h)^{\prod_i r_{i,t_i}}$ as $\text{MM}_{hk} \stackrel{\text{grp}}{=} e(c_1, \dots, c_k)^{a_X} e(B_X, h)$ with arbitrary a_X . Adjusting the probability for $a_X \neq 0$ in the order of $1/2q$ (as in the proof of Theorem 1) allows to prove (2) for $p(k) = 2 \cdot (2q)^n$. \square

Examples. For $R = \{0, 1\}$ and binary codes $\mathcal{C}_k : \{0, 1\}^k \rightarrow R^\ell$ with large minimum distance, we get the AHF implicit in [19]. This yields MPHFs that use $\mathbf{O}(k)$ groups \mathbb{G}_i , and have keys of $2k$ group elements. Larger R give new AHFs that yield MPHFs that use fewer groups, but have larger keys. For instance, with $R = \mathbb{F}_{2^\kappa}$, for $\kappa := \lfloor \log_2(k) \rfloor$, along with MDS codes over R , we obtain MPHFs that use $\mathbf{O}(k/\log_2(k))$ groups, and have keys consisting of k^2 group elements.

Theorem 4. *Let n be a constant, $q = q(k)$ be a polynomial, and let $\mathbf{H} = (\text{HGen}, \text{HEval})$ be a $(q + n - 1, 1)$ -MPHF into \mathbb{G}_ℓ . Then the group hash function $\mathbf{H}' = (\text{HGen}', \text{HEval}')$ with*

- $\text{HGen}'(1^k)$ that outputs $hk' = (hk_\nu)_{\nu \in [n]}$ for $hk_\nu \leftarrow \text{HGen}(1^k)$, and
- $\text{HEval}'(hk', X)$ that outputs $\mathbf{H}'_{hk'}(X) := \prod_{\nu \in [n]} \mathbf{H}_{hk_\nu}(X)$

is a (q, n) -MPHF into \mathbb{G}_ℓ .

Combining Theorems 3 and 4 yields a (poly, n) -MPHF for any constant n .

Proof. We construct suitable TGen' and TEval' algorithms from the respective TGen and TEval algorithms for \mathbf{H} :

- $\text{TGen}'(1^k, c_1, \dots, c_\ell, h)$ runs $(hk_\nu, td_\nu) \leftarrow \text{TGen}(1^k, c_1, \dots, c_\ell, h)$ for $\nu \in [n]$, and outputs $hk' := (hk_\nu)_{\nu \in [n]}$ and $td' := (td_\nu)_{\nu \in [n]}$.
- $\text{TEval}'(hk', X)$ invokes $(a_{\nu, X}, B_{\nu, X}) \leftarrow \text{TEval}(td_\nu, X)$ and outputs $a_X := \sum_{\nu \in [n]} a_{\nu, X}$ and $B_X := \prod_{\nu \in [n]} B_{\nu, X}$. This output can be justified with

$$\begin{aligned} \mathbf{H}'_{hk'}(X) &\stackrel{\text{grp}}{=} \prod_{\nu \in [n]} \mathbf{H}_{hk_\nu}(X) \stackrel{\text{grp}}{=} \prod_{\nu \in [n]} e(c_1, \dots, c_\ell)^{a_{\nu, X}} e(B_{\nu, X}, h) \\ &\stackrel{\text{grp}}{=} e(c_1, \dots, c_\ell)^{a_X} e(B_X, h). \end{aligned}$$

Now fix $X_1, \dots, X_q, Z_1, \dots, Z_n$ with $\{X_i\} \cap \{Z_j\} = \emptyset$. For each ν , we hope for the following event: $a_{\nu, X_i} = 0$ for all i , and $a_{\nu, Z_j} = 0$ exactly for $j \neq \nu$. For fixed ν , this event happens with probability at least $1/p(k)$ (over td_ν) for some polynomial p . Since $a_X = \sum_\nu a_{\nu, X}$, we get that with probability at least $(1/p(k))^n$, we have $a_{X_i} = 0$ for all i and $a_{Z_j} = a_{j, Z_j} \neq 0$ for all j . \square

4 (Hierarchical) ID-Based Non-interactive Key Exchange

Hierarchical identity-based non-interactive key exchange (H-ID-NIKE) is the natural generalisation of ID-NIKE [23, 11, 20] to the hierarchical setting: a root authority calculates and distributes private keys to sub-authorities, who in turn do the same for sub-sub-authorities, and so on, until leaf nodes are reached. Each node is identified by a vector of identities, and any pair of nodes in the tree should be able to non-interactively compute a common key based on their private keys and identities. We recall from the introduction that H-ID-NIKE schemes are rare, and, to the best of our knowledge, there are not even any ROM constructions that meet all the desirable criteria (efficiency, scalability, and full security in the sense of resilience to arbitrary node compromises).

Formally, an H-ID-NIKE scheme H-ID-NIKE consists of three PPT algorithms (see below), an identity space \mathcal{ID} and shared-key space \mathcal{SHK} . The users are organized in a tree of depth L whose root (at level 0) is the trusted authority (TA). The identity of a user at level $d \in [L]$ is represented by a vector $\mathbf{id} = (id_1, \dots, id_d) \in \mathcal{ID}^d$.

Setup. The setup algorithm $\text{Setup}(1^k, L)$ is run by the TA. Given the security parameter 1^k and a parameter $L \in \mathbb{N}$, it outputs a master public key mpk and a master secret key msk . We also interpret msk as the user secret key usk_ε for the empty identity ε .

Key Delegation. The key delegation algorithm $\text{Del}(mpk, usk_{\mathbf{id}}, \mathbf{id}')$ can be run by any user to generate a secret key for any of its children. Given the master public key mpk , the user secret key $usk_{\mathbf{id}}$ for an identity $\mathbf{id} = (id_1, \dots, id_d) \in \mathcal{ID}^d$, the algorithm outputs a user secret key $usk_{\mathbf{id}'}$ for any of its children $\mathbf{id}' = (id_1, \dots, id_\ell, id_{d+1}) \in \mathcal{ID}^{d+1}$ (for $0 \leq d < L$).

Shared Key Generation. Given the master public key mpk , a user secret key $usk_{\mathbf{id}_1}$ for an identity $\mathbf{id}_1 \in \mathcal{ID}^{\leq L}$, and an identity $\mathbf{id}_2 \in \mathcal{ID}^{\leq L}$, $\text{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$ outputs either a shared key $K_{\mathbf{id}_1, \mathbf{id}_2} \in \mathcal{SHK}$ or a failure symbol \perp . (If \mathbf{id}_1 is an ancestor of \mathbf{id}_2 (or vice-versa) the algorithm is assumed to always output \perp ¹⁴; here, \mathbf{id} is in particular considered to be an ancestor of itself. Otherwise the output is assumed to be in \mathcal{SHK} .)

For correctness, we require that for any $k, L \in \mathbb{N}$, for any $(mpk, msk) \leftarrow \text{Setup}(1^k, L)$, for any pair of identities $(\mathbf{id}_1, \mathbf{id}_2) \in \mathcal{ID}^{d_1} \times \mathcal{ID}^{d_2}$, such that neither is an ancestor of the other, and corresponding user secret keys $usk_{\mathbf{id}_1}$ and $usk_{\mathbf{id}_2}$ generated by repeated applications of Del from $usk_\varepsilon = msk$, we have $\text{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2) = \text{ShK}(mpk, usk_{\mathbf{id}_2}, \mathbf{id}_1)$.

A (non-hierarchical) ID-NIKE scheme is a H-ID-NIKE scheme in which the depth L of the tree is fixed to $L = 1$. (Note that in this case, Del gets as input $usk_\varepsilon = msk$ and outputs user secret keys for level-1 identities. We may thus also speak of extraction of user secret keys.)

¹⁴ If \mathbf{id}_1 is an ancestor of \mathbf{id}_2 , it can always compute the user secret key $usk_{\mathbf{id}_2}$; a key derived from $usk_{\mathbf{id}_2}$ can be used as a shared key between the two users.

4.1 Security Definition for (H-)ID-NIKE

We present a security model for H-ID-NIKE that is the natural generalisation of the *PS* model for ID-NIKE from [20] to the hierarchical setting. The model significantly strengthens the previous model of Gennaro *et al.* [14] by being fully adaptive, allowing arbitrary numbers of node corruptions, and allowing the adversary access to shared keys as well as user secret keys of inner (i.e., non-leaf) nodes. The model is defined in terms of a game between an adversary A and a challenger C . C takes as input the security parameter 1^k and a depth L , runs algorithm **Setup** of the H-ID-NIKE scheme and gives A the master public key mpk . It keeps the master secret key, msk , to itself. A then makes queries of the following three types:

Extract: A supplies an identity $\mathbf{id} = (id_1, \dots, id_d) \in \mathcal{ID}^d$ (for $d \in [L]$). C uses **Del** repeatedly, starting from msk , to derive $usk_{\mathbf{id}}$ and hands $usk_{\mathbf{id}}$ to A .

Reveal: Here A supplies a pair $(\mathbf{id}_1, \mathbf{id}_2) \in \mathcal{ID}^{d_1} \times \mathcal{ID}^{d_2}$. C extracts $usk_{\mathbf{id}_1}$ as above, runs $K_{\mathbf{id}_1, \mathbf{id}_2} \leftarrow \text{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$, and hands $K_{\mathbf{id}_1, \mathbf{id}_2}$ to A .

Test: A supplies two *target* identities $(\mathbf{id}_1^*, \mathbf{id}_2^*) \in \mathcal{ID}^{d_1} \times \mathcal{ID}^{d_2}$ such that neither is an ancestor of the other. C computes $K_{\mathbf{id}_1^*, \mathbf{id}_2^*}$ as above, and tosses a coin $b \leftarrow \{0, 1\}$. If $b = 0$ then C gives $K_{\mathbf{id}_1^*, \mathbf{id}_2^*}$ to A ; otherwise, if $b = 1$, then C gives A a uniform element from \mathcal{SHK} .

Finally, A outputs a guess \hat{b} for b . In our security model, the adversary is allowed to make an arbitrary (but polynomial) number of **Extract** and **Reveal** queries. Furthermore, the adversary is fully adaptive, in the sense that it can compromise nodes (by making **Extract** and/or **Reveal** queries) in any order. In order to prevent the adversary from trivially winning, we require that the adversary is not allowed to make any **Extract** queries on an ancestor of \mathbf{id}_1^* or \mathbf{id}_2^* , and no **Reveal** query on the pairs $(\mathbf{id}_1^*, \mathbf{id}_2^*)$ and $(\mathbf{id}_2^*, \mathbf{id}_1^*)$. The advantage of an adversary A against a H-ID-NIKE scheme H-ID-NIKE is

$$\begin{aligned} \text{Adv}_{A, \text{H-ID-NIKE}}^{\text{IND-SK}}(k) &= \left| \Pr[\hat{b} = b] - 1/2 \right| \\ &= 1/2 \left| \Pr[\hat{b} = 1 \mid b = 1] - \Pr[\hat{b} = 1 \mid b = 0] \right|. \end{aligned}$$

We say that H-ID-NIKE is IND-SK secure iff $\text{Adv}_{A, \text{H-ID-NIKE}}^{\text{IND-SK}}(k)$ is negligible for all PPT adversaries A .

In the non-hierarchical case (i.e., $L = 1$), we recover the definition and security model for (non-hierarchical) ID-NIKE from [20]. Note also that versions of these models in which multiple **Test** queries are permitted for a single bit b can be shown to be polynomially equivalent to the versions with a single **Test** query using standard hybrid arguments.

4.2 Fully-Secure ID-NIKE from MPHFs

In this section we revisit the ID-NIKE scheme of Sakai, Ohgishi and Kasahara (SOK) [23]. We replace random oracles with (poly, 2)-MPHFs in their scheme

and prove security of the generalized scheme. Using our standard-model MPHFs, this yields the first standard-model ID-NIKE scheme.¹⁵ We then consider a hierarchical generalisation.

We assume a 2ℓ -group system $\mathcal{MPG}_{2\ell} = \{\{\mathbb{G}_i\}_{i \in [2\ell]}, p, \{e_{i,j}\}_{i,j \geq 1, i+j \leq 2\ell}\}$ generated by a multilinear maps parameter generator $\mathcal{MG}_{2\ell}(1^k)$, and a $(\text{poly}, 2)$ -MPHF $H = (\text{HGen}, \text{HEval})$ with input length in $\{0, 1\}^k$ and output in \mathbb{G}_ℓ . The component algorithms of our ID-NIKE scheme $\text{IDNIKE}_{\text{MPHF}}$ are then defined in Figure 1. (For compatibility with existing notation, we present an extraction algorithm Ext instead of an equivalent delegation algorithm.) Correctness of the scheme is easy to verify.

<p>Algorithm Setup(1^k) $\mathcal{MPG}_{2\ell} \leftarrow \mathcal{MG}_{2\ell}(1^k)$ $x \leftarrow \mathbb{Z}_p, hk \leftarrow \text{HGen}(1^k)$ $mpk := (\mathcal{MPG}_{2\ell}, hk), msk := x$ return (mpk, msk)</p>	<p>Algorithm Ext(mpk, msk, id) $usk_{id} \leftarrow \text{reRand}_3(\text{H}_{hk}(id)^{msk})$ return usk_{id}</p> <hr style="border: 0.5px solid black;"/> <p>Algorithm ShK(mpk, usk_{id_1}, id_2) $K_{id_1, id_2} := \text{ext}(e(usk_{id_1}, \text{H}_{hk}(id_2)))$ return K_{id_1, id_2}</p>
---	---

Fig. 1. The ID-NIKE scheme $\text{IDNIKE}_{\text{MPHF}}$

Theorem 5 (Security of the MPHF-based ID-NIKE scheme). *Assume H is a $(\text{poly}, 2)$ -MPHF into \mathbb{G}_ℓ . Then $\text{IDNIKE}_{\text{MPHF}}$ is IND-SK secure under the $(2\ell + 1)$ -power assumption.*

Proof. See the full version (<http://eprint.iacr.org/2013/354.pdf>).

A Variant Secure under a Weaker Assumption. We can also construct an ID-NIKE scheme in the standard model using two instances (with keys hk_1, hk_2) of a $(\text{poly}, 1)$ -MPHF instead of a single instance of a $(\text{poly}, 2)$ -MPHF. Shared keys are computed as $K := \text{ext}(e(\text{H}_{hk_1}(id_1)^{msk}, \text{H}_{hk_2}(id_2)))$; user secret keys are of the form $usk_{id} = (\text{reRand}_3(\text{H}_{hk_1}(id)^{msk}), \text{reRand}_3(\text{H}_{hk_2}(id)^{msk}))$. The benefit of this variant is that it is possible to prove security under the 2ℓ -MDDH assumption (as opposed to the potentially stronger $(2\ell + 1)$ -power assumption we use above).

4.3 Extension to H-ID-NIKE

We can extend our ID-NIKE scheme to a H-ID-NIKE scheme of constant depth L . To this end, we work in a $2\ell L$ -group system $\mathcal{MPG}_{2\ell L}$, and use L instances of a $(\text{poly}, 2)$ -MPHF H into \mathbb{G}_ℓ . The resulting H-ID-NIKE scheme, denoted by

¹⁵ If we instantiate the MPHFs again with random oracles (using Theorem 1), we retrieve the original SOK scheme in pairing-friendly groups, along with a security proof. However, we note that our security proof uses a different, seemingly stronger computational assumption.

HIDNIKE_{MPHF}, is given in Figure 2. In that description, and in the following, we write $\mathbf{id}_{\lceil i} := (id_1, \dots, id_i)$ for an identity $\mathbf{id} = (id_1, \dots, id_d)$ and $i \leq d$. We assume that all involved identities (including “shortened identities” $\mathbf{id}_{\lceil i}$) can be uniquely encoded as k -bit strings. (If this is not the case, we can always first apply a collision-resistant hash function.)

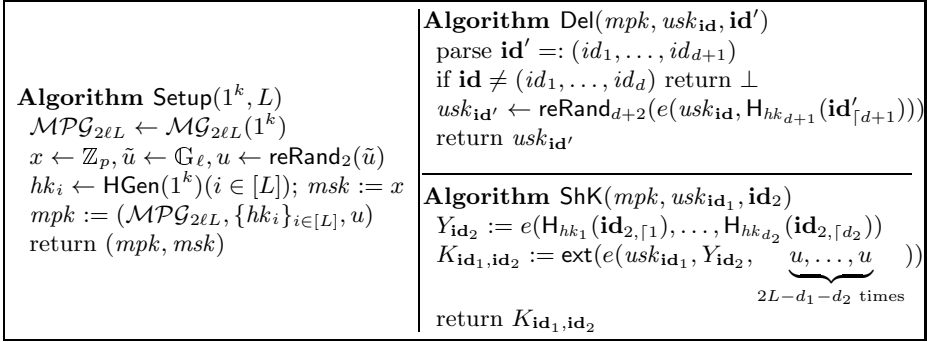


Fig. 2. The H-ID-NIKE scheme HIDNIKE_{MPHF}

Note. $msk = usk_\varepsilon = x \in \mathbb{Z}_p = \mathbb{G}_0$, so Del can be used to derive level-1 user secret keys from msk . (Recall that our definition of e is consistent with the implicit exponent group $\mathbb{G}_0 = \mathbb{Z}_p$; e.g., $e(x, g) = g^x$ for $x \in \mathbb{G}_0$.)

Theorem 6 (Security of the MPHF-based H-ID-NIKE scheme). *Let H be a (poly, 2)-MPHF into \mathbb{G}_ℓ . For fixed depth $L \in \mathbb{N}$, HIDNIKE_{MPHF} is secure under the $(2\ell L + 1)$ -power assumption.*

Proof. See the full version (<http://eprint.iacr.org/2013/354.pdf>).

A More Efficient Variant in the Random Oracle Model. We can replace the $2\ell L$ -group system with a $2L$ -group system and the L different MPHF’s with a random oracle hashing into \mathbb{G}_1 in the above scheme HIDNIKE_{MPHF} to obtain a second H-ID-NIKE scheme which can be proven secure in the random oracle model. In this case, the $2L$ -group system can be instantiated with smaller parameters than the $2\ell L$ -group system required in our standard model scheme.

Security with Multiple TAs and Group-ID-NIKE. We can also achieve security in the more general setting of multiple trusted authorities and shared keys that can be computed by groups of parties instead of just pairs. The details can be found in the full version.

5 IBE and Signature Schemes from MPHF’s

Identity-Based Encryption. An identity-based encryption (IBE) scheme IBE with identity space \mathcal{ID} and message space \mathcal{M} consists of four PPT algorithms:

Gen, Ext, Enc, Dec. Key generation $\text{Gen}(1^k)$, on input a security parameter 1^k , outputs a master public key mpk and a master secret key msk . Key extraction $\text{Ext}(msk, id)$, given msk and an identity $id \in \mathcal{ID}$, outputs a user secret key usk_{id} . Encryption $\text{Enc}(mpk, id, M)$, given mpk , an identity $id \in \mathcal{ID}$, and a message $M \in \mathcal{M}$, outputs a ciphertext C . Decryption $\text{Dec}(usk_{id}, C)$, given usk_{id} and a ciphertext C , outputs a message $M \in \mathcal{M} \cup \{\perp\}$. For correctness, we require that for any $k \in \mathbb{N}$, all $(mpk, msk) \leftarrow \text{Gen}(1^k)$, all $id \in \mathcal{ID}$, all $usk_{id} \leftarrow \text{Ext}(msk, id)$, all $M \in \mathcal{M}$, and all $C \leftarrow \text{Enc}(mpk, id, M)$, Dec satisfies $\text{Dec}(usk_{id}, C) = M$.

IBE-IND-CPA Security. An IBE scheme IBE as above is IBE-IND-CPA secure iff every PPT adversary A succeeds in the following experiment with probability at most negligibly larger than $1/2$. First, A gets an honestly generated master public key mpk ; in all of the following, A has access to an $\text{Ext}(msk, \cdot)$ oracle for the corresponding msk . Next, A selects an identity $id^* \in \mathcal{ID}$ and two equal-length messages $M_0, M_1 \in \mathcal{M}$. The experiment then computes $C^* \leftarrow \text{Enc}(mpk, id^*, M_b)$ for uniformly chosen $b \leftarrow \{0, 1\}$ and sends C^* to A . Finally, A outputs a guess b' and succeeds iff $b = b'$ and it has not queried Ext with id^* .

IBE from (poly, 1)-MPHFs. Figure 3 depicts IBE_{MPHF} , which is the Boneh-Franklin IBE scheme [4], implemented with (poly, 1)-MPHFs. Message and identity space are $\mathcal{M} = \mathcal{ID} = \{0, 1\}^k$. We assume an $(\ell+1)$ -group system $\mathcal{MPG}_{\ell+1} = \{\{\mathbb{G}_i\}_{i \in [\ell+1]}, p, \{e_{i,j}\}_{i,j \geq 1, i+j \leq \ell+1}\}$ generated by a multilinear maps parameter generator $\mathcal{MG}_{\ell+1}(1^k)$, and a (poly, 1)-MPHF H into \mathbb{G}_ℓ . If we take a random oracle as (poly, 1)-MPHF (as in Theorem 1), then $\ell = 1$, and we get the original BF scheme. Correctness of IBE_{MPHF} is easy to verify.

<p>Algorithm $\text{Gen}(1^k)$ $\mathcal{MPG}_{\ell+1} \leftarrow \mathcal{MG}_{\ell+1}(1^k)$ $hk \leftarrow \text{HGen}(1^k), h \leftarrow \mathbb{G}_1, x \leftarrow \mathbb{Z}_p$ $mpk := (\mathcal{MPG}_{\ell+1}, hk, h, \text{reRand}_2(h^x))$ $msk := (hk, x)$ return (mpk, msk)</p>	<p>Algorithm $\text{Enc}(mpk, id, M)$ parse $mpk =: (\mathcal{MPG}_{\ell+1}, hk, h, \tilde{h})$ $r \leftarrow \mathbb{Z}_p$ $C :=$ $(\text{reRand}_2(h_1^r), \text{ext}(e(H_{hk}(id), \tilde{h})^r) \oplus M)$ return C</p>
<p>Algorithm $\text{Ext}(msk, id)$ parse $msk =: (hk, x)$ return $usk_{id} := \text{reRand}_3(H_{hk}(id)^x)$</p>	<p>Algorithm $\text{Dec}(usk_{id}, C)$ parse $C =: (C_1, C_2)$ return $M := C_2 \oplus \text{ext}(e(usk_{id}, C_1))$</p>

Fig. 3. The IBE scheme from (poly, 1)-MPHFs

Theorem 7. Assume IBE_{MPHF} is implemented in an $(\ell + 1)$ -group system, and with a (poly, 1)-MPHF H into \mathbb{G}_ℓ . Then, under the $(\ell + 1)$ -MDDH assumption, IBE_{MPHF} is IBE-IND-CPA-secure.

Proof. See the full version (<http://eprint.iacr.org/2013/354.pdf>).

Extension to HIBE. We can extend the above IBE scheme to a hierarchical IBE (HIBE) scheme of constant depth D . This generalization works similarly as in the ID-NIKE case. Hence, due to space constraints, we postpone a more detailed exposition to the full version of this paper.

(Hierarchical) Signatures from (poly, 1)-MPHF. We can convert any (H)IBE scheme into a (hierarchical) signature scheme using the techniques of [4, 15, 10]. If we apply this transformation to IBE_{MPHF} above, we obtain an abstraction of BLS signatures [7]. Indeed, if we instantiate the involved MPHF with a random oracle, we get the original BLS scheme. On the other hand, if we use the standard-model MPHF from Theorem 3, we obtain (a slight variant of) the signature scheme of Boneh and Silverberg [5]. In fact, with suitable parameters (i.e., a larger R , see Section 3.5), we obtain a signature scheme that uses only $\mathbf{O}(k/\log(k))$ groups and multilinear operations (as opposed to $\mathbf{O}(k)$ groups and multilinear operations in the Boneh-Silverberg scheme). It seems natural to expect that, using the techniques of [22], this also yields an aggregatable signature scheme.

References

- [1] Abdalla, M., Fiore, D., Lyubashevsky, V.: From selective to full security: Semi-generic transformations in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 316–333. Springer, Heidelberg (2012)
- [2] Blundo, C., Santis, A.D., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly secure key distribution for dynamic conferences. *Inf. Comput.* 146(1), 1–23 (1998)
- [3] Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
- [4] Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 213. Springer, Heidelberg (2001)
- [5] Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. *Contemporary Mathematics* 324, 71–90 (2002)
- [6] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
- [7] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. *J. Cryptology* 17(4), 297–319 (2004)
- [8] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
- [9] Çapar, Ç., Goeckel, D., Paterson, K.G., Quaglia, E.A., Towsley, D., Zafer, M.: Signal-flow-based analysis of wireless security protocols. *Information and Computation* 226, 37–56 (2013)
- [10] Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for signatures from identity-based encryption. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 218–227. Springer, Heidelberg (2007)

- [11] Dupont, R., Enge, A.: Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics* 154(2), 270–276 (2006)
- [12] Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (2013)
- [13] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013), <http://eprint.iacr.org/2012/610>
- [14] Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T., Reidt, S., Wolthusen, S.D.: Strongly-resilient and non-interactive hierarchical key-agreement in MANETs. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 49–65. Springer, Heidelberg (2008)
- [15] Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
- [16] Guo, H., Mu, Y., Li, Z., Zhang, X.: An efficient and non-interactive hierarchical key agreement protocol. *Computers & Security* 30(1), 28–34 (2011)
- [17] Hanaoka, G., Matsuda, T., Schuldt, J.C.N.: On the impossibility of constructing efficient key encapsulation and programmable hash functions in prime order groups. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 812–831. Springer, Heidelberg (2012)
- [18] Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
- [19] Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 597–612. Springer, Heidelberg (2002)
- [20] Paterson, K.G., Srinivasan, S.: On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography* 52(2), 219–241 (2009)
- [21] Ramkumar, M., Memon, N., Simha, R.: A hierarchical key pre-distribution scheme. In: 2005 IEEE International Conference on Electro Information Technology (May 2005)
- [22] Rückert, M., Schröder, D.: Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In: Park, J.H., Chen, H.-H., Atiquz-zaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 750–759. Springer, Heidelberg (2009)
- [23] Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000, Okinawa, Japan (January 2000)
- [24] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)