Programmable Hash Functions in the Multilinear Setting ⋆

Eduarda S.V. Freire[1,⋆⋆], Dennis Hofheinz[2,⋆⋆⋆], Kenneth G. Paterson[1,†] and Christoph Striecks[2]

[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

⋆ This is the full version of a paper with the same title accepted to CRYPTO 2013, © IACR 2013.
⋆⋆ 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.

X (e.g., a message to be signed) to a group element $\mathsf{H}(X)$; a special trapdoor allows to decompose $\mathsf{H}(X) = c^{a_X} h^{b_X}$ for previously chosen c, h. In a larger proof, c will usually be a "challenge element" (e.g., a part of a given Diffie-Hellman challenge), so that $\mathsf{H}(X)$ 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

[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 $Z_1, \ldots, Z_n$ (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.

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.

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

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

(where X_i denotes the i-th bit of X) evaluates to zero everywhere except for $X = (b_1, \ldots, b_{|X|})$. In fact, we implement a suitable variant of the function in (1) in the exponent (in the sense that $\mathsf{H}(X) = c^{a_X} h^{b_X} = c^{\alpha(X)} h^{b_X}$ 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

[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.

this already hints at the potential of our notion of PHFs, their actual usefulness 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. We also show how multilinear maps can be used

[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.

to achieve security in the broader scenario of multiple TAs, and for shared keys among whole groups of parties.

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 $[n] := \{1, \ldots, \lfloor n \rfloor\}$. 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 $\mathcal{S}^1, \mathcal{S}^2, \ldots$ 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 $f : \mathbb{N} \to \mathbb{R}$ 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 $\mathbb{G}_1, \ldots, \mathbb{G}_\ell$ of prime order p, along with bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$ 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 $e(h_1, \ldots, h_j) := e(h_1, e(h_2, \ldots, e(h_{j-1}, h_j) \ldots))$ for $h_j \in \mathbb{G}_{i_j}$ and $i = (i_1 + i_2 + \ldots + i_j) \leq \ell$. 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* $\mathcal{MG}_\ell$ 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]

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.
- $g \overset{\mathsf{enc}}{=} h$ holds iff the encodings g and h match.
- $g \overset{\mathsf{grp}}{=} h$ 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]
- $\mathsf{reRand}_j(g)$ 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 $\mathsf{reRand}_j(g)$ and $\mathsf{reRand}_j(h)$ are statistically close for any two encodings g, h with $g \overset{\mathsf{grp}}{=} h$ and noise level less than j.
- $\mathsf{ext}(g)$ denotes the canonical bitstring extracted from encoding g. We have $\mathsf{ext}(g) = \mathsf{ext}(h)$ for any g, h with $g \overset{\mathsf{grp}}{=} h$ 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 $(g, g^{x_1}, \ldots, g^{x_{\ell+1}})$, (for $g \leftarrow \mathbb{G}_1$ and uniform exponents x_i), the element $e(g^{x_1}, \ldots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniform $\mathbb{G}_\ell$-element. The $(\ell+1)$-power assumption is: given (g, g^x) (for $g \leftarrow \mathbb{G}_1$ and uniform x), the element $e(\underbrace{g^x, \ldots, g^x}_{\ell \text{ times}})^x \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniformly chosen $\mathbb{G}_\ell$-element.[8]

[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.

[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 $e(g^{x_1}, \ldots, g^{x_\ell})^{x_{\ell+1}})$ (resp. $e(g^x, \ldots, g^x)^x$) are produced with knowledge of the exponents x, x_i as fully randomized but low-noise encodings.

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). Namely, the image $\mathsf{H}(X)$ of a PHF can always be explained as $\mathsf{H}(X) = c^{a_X} h^{b_X}$ 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 $\mathsf{H}(X_i)$ contain no challenge component, while all $\mathsf{H}(Z_j)$ 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 $\mathbb{G}_\ell$. Here instead of explaining $\mathsf{H}(X)$ as a product $c^{a_X} h^{b_X}$ for c, h in the target group $\mathbb{G}_\ell$ (as the case of PHFs), we will explain $\mathsf{H}(X)$ as a product $e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$, for externally given challenges $c_i \in \mathbb{G}_1$ (which means $c = e(c_1, \ldots, c_\ell) \in \mathbb{G}_\ell$) 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* $\mathsf{HGen}(1^k)$ *outputs a key* hk*, and* $\mathsf{HEval}(hk, X)$ *(for a key* hk *and* $X \in \{0,1\}^k$*) deterministically outputs an image* $\mathsf{H}_{hk}(X) \in \mathbb{G}$*.*

Definition 2 (MPHF). *Assume an* ℓ'*-group system* $\mathcal{MPG}_{\ell'}$ *as generated by* $\mathcal{MG}_{\ell'}(1^k)$*. Let* H *be a group hash function into* $\mathbb{G}_\ell$ *(*$\ell \leq \ell'$*), 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* TGen *and* TEval *as follows.*

- $\mathsf{TGen}(1^k, c_1, \ldots, c_\ell, h)$ *(for* $c_i, h \in \mathbb{G}_1$ *and* $h \overset{\mathsf{grp}}{\neq} 1$*) 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* HGen*.*[9]

[9] There is a subtlety here: in case of encoded group elements, the output of TGen 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.

– $\mathsf{TEval}(td, X)$ *(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* $\mathsf{H}_{hk}(X) \stackrel{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} \cdot e(B_X, h)$*. We require that there is a polynomial* $p(k)$ *such that for all* hk *and* $X_1, \ldots, X_m$*,* $Z_1, \ldots, Z_n \in \{0,1\}^k$ *with* $\{X_i\}_i \cap \{Z_j\}_j = \emptyset$*,*

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

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 $\mathsf{RO}(X) := c^{a_X} h^{B_X}$ 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 $X_1, \ldots, X_q, Z_1, \ldots, Z_n$, we thus have

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

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 $(\mathsf{poly}, 1)$-(M)PHFs. An AHF input X is mapped to an image $\mathsf{AHF}(X)$ in a way that X can fall in the set of controlled, $\mathtt{CO}$, inputs (meaning that we know a trapdoor that allows to answer adversary's queries for that input) or uncontrolled, $\mathtt{UN}$, 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 $X_1, \ldots, X_q, Z$, 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 $\mathsf{AHF} : \{0,1\}^k \to R^\ell$ (with a finite set[10] R and polynomial $\ell = \ell(k)$) and $K \in (R \cup \{\perp\})^\ell$, define the function $F_K : \{0,1\}^k \to \{\mathtt{CO}, \mathtt{UN}\}$ through $F_K(X) = \mathtt{UN} \iff \forall i : K_i = \mathsf{AHF}(X)_i \vee K_i = \perp$, where $\mathsf{AHF}(X)_i$ denotes the i-th component of $\mathsf{AHF}(X)$.[11] We say that AHF is q-admissible if there exists a PPT algorithm KGen and a polynomial $p(k)$, such that for all $X_1, \ldots, X_q, Z \in \{0,1\}^k$ with $Z \notin \{X_i\}$,*

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

where the probability is over $K \leftarrow \mathsf{KGen}(1^k)$. 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., $F_K(X) = \mathtt{CO}$) if there is an i with $X_i \neq K_i \neq \perp$.

Theorem 2 ([19]). *Assume a family of codes $\{\mathcal{C}_k\}$ with $\mathcal{C}_k : \{0,1\}^k \to R^\ell$ denoting both the code and its encoding function. Suppose that $\mathcal{C}_k$ 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 $\mathcal{C}_k(X_1)$ and $\mathcal{C}_k(X_2)$ differ in $\geq c \cdot \ell$ positions.) Then $\{\mathcal{C}_k\}$ is an AHF.*

Proof. Let $q = q(k)$ be a polynomial. We need to devise a PPT algorithm KGen such that (3) holds. $\mathsf{KGen}(1^k)$ 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 $X_1, \ldots, X_q, Z \in \{0,1\}^k$ with $Z \notin \{X_i\}$. Our choice of K implies $\Pr[F_K(Z) = \mathtt{UN}] = |R|^{-d}$. For any fixed i, we want to upper bound the probability $\Pr[F_K(X_i) = \mathtt{UN} \mid F_K(Z) = \mathtt{UN}]$. (This step loosely corresponds to [19,

[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 $F_K(X) = \mathtt{CO}$ iff there is an i with $K_i = 1 - \mathsf{AHF}(X)_i$.

Lemma 4].) Hence, assume $F_K(Z) = \mathtt{UN}$; note that this conditioning leaves the distribution of I uniform. Now $\mathcal{C}_k(X_i)$ and $\mathcal{C}_k(Z)$ differ in a set $\Delta \subseteq [\ell]$ of positions with $|\Delta| \geq c\ell$. Hence, $F_K(X_i) = \mathtt{UN}$ is equivalent to $I \cap \Delta = \emptyset$. Thus,

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

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

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

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

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

Theorem 3. *The group hash function* MM *above is a* $(\mathsf{poly}, 1)$*-MPHF.*

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

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

For now, assume $c_i \overset{\mathsf{grp}}{\neq} 1$ 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 $td := ((c_i), h, K, (r_{i,j}))$.

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

[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 $F_K(X) = \texttt{CO}$**,** 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

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

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

Case $F_K(X) = \texttt{UN}$**,** i.e., $K_i = t_i$ or $K_i = \perp$ for all i. This means that $h_{i,t_i} \overset{\mathsf{grp}}{=} c_i^{r_{i,t_i}}$ for all i, so $\mathsf{MM}_{hk}(X) \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$ 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 $c_i \overset{\mathsf{grp}}{=} 1$ for some i, we have $e(c_1, \ldots, c_\ell) \overset{\mathsf{grp}}{=} 1$. If we replace *all* c_i in (4) with h, we can explain any image $\mathsf{MM}_{hk}(X) \overset{\mathsf{grp}}{=} e(h, \ldots, h)^{\prod_i r_{i,t_i}}$ as $\mathsf{MM}_{hk} \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_k)^{a_X} e(B_X, h)$ 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 $p(k) = 2 \cdot (2q)^n$. □

Examples. For $R = \{0,1\}$ and binary codes $\mathcal{C}_k : \{0,1\}^k \to R^\ell$ 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 $\mathsf{H} = (\mathsf{HGen}, \mathsf{HEval})$ be a $(q + n - 1, 1)$-MPHF into $\mathbb{G}_\ell$. Then the group hash function $\mathsf{H}' = (\mathsf{HGen}', \mathsf{HEval}')$ with*

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

is a (q, n)-MPHF into $\mathbb{G}_\ell$.

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 H:

- $\mathsf{TGen}'(1^k, c_1, \ldots, c_\ell, h)$ runs $(hk_\nu, td_\nu) \leftarrow \mathsf{TGen}(1^k, c_1, \ldots, c_\ell, h)$ for $\nu \in [n]$, and outputs $hk' := (hk_\nu)_{\nu \in [n]}$ and $td' := (td_\nu)_{\nu \in [n]}$.
- $\mathsf{TEval}'(hk', X)$ invokes $(a_{\nu,X}, B_{\nu,X})) \leftarrow \mathsf{TEval}(td_\nu, X)$ 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

$$\mathsf{H}'_{hk'}(X) \overset{\mathsf{grp}}{=} \prod_{\nu \in [n]} \mathsf{H}_{hk_\nu}(X) \overset{\mathsf{grp}}{=} \prod_{\nu \in [n]} e(c_1, \ldots, c_\ell)^{a_{\nu,X}} e(B_{\nu,X}, h)$$
$$\overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h).$$

Now fix $X_1, \ldots, X_q, Z_1, \ldots, Z_n$ 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. □

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 $\mathsf{H\text{-}ID\text{-}NIKE}$ 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 $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$.

Setup. The setup algorithm $\mathsf{Setup}(1^k, L)$ 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 $\mathsf{Del}(mpk, usk_{\mathbf{id}}, \mathbf{id}')$ 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 $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$, the algorithm outputs a user secret key $usk_{\mathbf{id}'}$ for any of its children $\mathbf{id}' = (id_1, \ldots, id_\ell, id_{d+1}) \in \mathcal{ID}^{d+1}$ (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 $\mathbf{id}_2 \in \mathcal{ID}^{\leq L}$, $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$ 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 $(mpk, msk) \leftarrow \mathsf{Setup}(1^k, L)$, 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 $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2) = \mathsf{ShK}(mpk, usk_{\mathbf{id}_2}, \mathbf{id}_1)$.

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

[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.

$usk_\varepsilon = msk$ and outputs user secret keys for level-1 identities. We may thus also speak of extraction of user secret keys.)

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 $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$ (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 $K_{\mathbf{id}_1,\mathbf{id}_2} \leftarrow \mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$, 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 $\mathsf{H\text{-}ID\text{-}NIKE}$ is

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

We say that $\mathsf{H\text{-}ID\text{-}NIKE}$ is IND-SK secure iff $\mathrm{Adv}^{\text{IND-SK}}_{A,\mathsf{H\text{-}ID\text{-}NIKE}}(k)$ 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 $(\mathsf{poly}, 2)$-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 $(\mathsf{poly}, 2)$-MPHF $\mathsf{H} = (\mathsf{HGen}, \mathsf{HEval})$ with input length in $\{0,1\}^k$ and output in $\mathbb{G}_\ell$. The component algorithms of our ID-NIKE scheme $\mathsf{IDNIKE}_{\mathsf{MPHF}}$ 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. We now prove security.

Algorithm $\mathsf{Setup}(1^k)$
$\mathcal{MPG}_{2\ell} \leftarrow \mathcal{MG}_{2\ell}(1^k)$
$x \leftarrow \mathbb{Z}_p, hk \leftarrow \mathsf{HGen}(1^k)$
$mpk := (\mathcal{MPG}_{2\ell}, hk), msk := x$
return (mpk, msk)

Algorithm $\mathsf{Ext}(mpk, msk, id)$
$usk_{id} \leftarrow \mathsf{reRand}_3(\mathsf{H}_{hk}(id)^{msk})$
return usk_{id}

Algorithm $\mathsf{ShK}(mpk, usk_{id_1}, id_2)$
$K_{id_1,id_2} := \mathsf{ext}(e(usk_{id_1}, \mathsf{H}_{hk}(id_2)))$
return K_{id_1,id_2}

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

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

Proof. Assume an IND-SK adversary A against $\mathsf{IDNIKE}_{\mathsf{MPHF}}$. We construct a $(2\ell+1)$-power distinguisher B that, given a 2ℓ-group system $\mathcal{MPG}_{2\ell}$, and group elements $g, g^x \in \mathbb{G}_1$ and $S \in \mathbb{G}_{2\ell}$, distinguishes between $S \stackrel{\mathsf{grp}}{=} e(g,\ldots,g)^{x^{2\ell+1}}$ (i.e., S is real), and random S.

Concretely, B will internally simulate A, together with the IND-SK experiment. Let id_1^*, id_2^* be the identities from A's ***Test*** query. Furthermore, let $q = q(k)$ be a polynomial upper bound on the total number of identities $id_i \notin \{id_1^*, id_2^*\}$ that appear in A's ***Extract*** and ***Reveal*** queries. In the following, we will use the $(q,2)$-MPHF property of H (and the corresponding algorithms TGen and TEval). B first runs $(hk, td) \leftarrow \mathsf{TGen}(1^k, g^x, \ldots, g^x, g)$ and and sets $mpk := (\mathcal{MPG}_{2\ell}, hk)$. Implicitly, we will have $msk := x$.

[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.

We will first describe how B answers an $\boldsymbol{Extract}(id)$ query of A. If $a_{id} = 0$ (for $(a_{id}, B_{id}) := \mathsf{TEval}(td, id)$), then B can compute $usk_{id} \leftarrow \mathsf{reRand}_3(e(B_{id}, g^x)) \stackrel{\mathsf{grp}}{=} \mathsf{H}_{hk}(id)^{msk}$. Otherwise, B aborts with output 0. We will hope for the event that $a_{id_i} = 0$ for all q identities $id_i \notin \{id_1^*, id_2^*\}$ from B's $\boldsymbol{Extract}$ and $\boldsymbol{Reveal}$ queries. In that case, B can answer not only all $\boldsymbol{Extract}$ queries from A, but also all $\boldsymbol{Reveal}$ queries (by first computing the user secret key usk_{id} of one of the two involved identities, and then using usk_{id} to compute the shared key).

We will additionally hope for $a_{id_1^*}, a_{id_2^*} \neq 0$; in this case, B can embed its own challenge into the reply K^* to A's $\boldsymbol{Test}$ query as

$$K^* := \mathsf{ext}(S^{a_{id_1^*} a_{id_2^*}} \cdot e(B_{id_1^*}, B_{id_2^*}, g, g^x) \cdot e(B_{id_1^*}, \underbrace{g^x, \ldots, g^x}_{\ell+1 \text{ times}})^{a_{id_2^*}} \cdot e(B_{id_2^*}, \underbrace{g^x, \ldots, g^x}_{\ell+1 \text{ times}})^{a_{id_1^*}}). \quad (5)$$

By using $\mathsf{H}_{hk}(id_i^*) \stackrel{\mathsf{grp}}{=} e(\overbrace{g^x, \ldots, g^x}^{\ell \text{ times}})^{a_{id_i^*}} e(B_{id_i^*}, g)$, we see that $K^* = \mathsf{ext}(e(\mathsf{H}_{hk}(id_1^*)^x, \mathsf{H}_{hk}(id_2^*))) = K_{id_1^*, id_2^*}$ whenever $S \stackrel{\mathsf{grp}}{=} e(g^x, \ldots, g^x)^x$. Conversely, if S is random, then so is K^*. (If $a_{id_i^*} = 0$ for some $i \in \{1, 2\}$, then B aborts with output 0.)

Finally, B outputs $\hat{b}$ (i.e., A's guess for the bit b in the IND-SK experiment). If the event that B aborts is independent of the queried identities id_i and id_i^* (as is the case for the RO-based MPHF from Theorem 1), we have

$$|\Pr[B = 1 \mid S \text{ real}] - \Pr[B = 1 \mid S \text{ random}]| = \Pr[\neg\mathsf{abort}] \cdot \mathrm{Adv}^{\text{IND-SK}}_{A, \mathsf{IDNIKE}_{\mathsf{MPHF}}}(k)/2.$$

Hence, B breaks the $(2\ell+1)$-power assumption iff A breaks the IND-SK security of $\mathsf{IDNIKE}_{\mathsf{MPHF}}$.

However, in the general case, abort might not be independent of the id_i and id_i^*. Hence, we will have to resort to an "artificial abort" strategy as in [24]. That is, even if $a_{id_i} = 0$ and $a_{id_i^*} \neq 0$ for all i, B will "artificially" abort with probability $1 - 1/(P_{(id_i),(id_i^*)} \cdot p(k))$ for the polynomial $p(k)$ from (3) and $P_{(id_i),(id_i^*)} := \Pr[\neg\mathsf{abort} \mid (id_i), (id_i^*)]$. This keeps the (new) abort probability at $1/p(k)$, independently of the id_i and id_i^*, and enables an analysis as above. Unfortunately, in the general case, we can only approximate $P_{(id_i),(id_i^*)}$ (up to an inversely polynomial error, by running TEval with freshly generated keys sufficiently often), which introduces an additional error term in the analysis. We refer to [24] for details on the artificial abort technique. □

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 $(\mathsf{poly}, 1)$-MPHF instead of a single instance of a $(\mathsf{poly}, 2)$-MPHF. Shared keys are computed as $K := \mathsf{ext}(e(\mathsf{H}_{hk_1}(id_1)^{msk}, \mathsf{H}_{hk_2}(id_2)))$; user secret keys are of the form $usk_{id} = (\mathsf{reRand}_3(\mathsf{H}_{hk_1}(id)^{msk}), \mathsf{reRand}_3(\mathsf{H}_{hk_2}(id)^{msk}))$. 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). The proof is similar to the one above; however, we will hope that $a_{1,id_i} = a_{2,id_i} = 0$ for

all non-challenge queries id_i, and that $a_{1,id_1^*}, a_{2,id_2^*} \neq 0$ and $a_{1,id_2^*} = a_{2,id_1^*} = 0$, where $(a_{j,id}, B_{j,id}) = \mathsf{TEval}(td_j, id)$.

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 $(\mathsf{poly}, 2)$-MPHF H into $\mathbb{G}_\ell$. The resulting H-ID-NIKE scheme, denoted by $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$, is given in Figure 2. In that description, and in the following, we write $\mathbf{id}_{\lceil i} := (id_1, \ldots, id_i)$ for an identity $\mathbf{id} = (id_1, \ldots, id_d)$ 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.)

Algorithm $\mathsf{Setup}(1^k, L)$
$\mathcal{MPG}_{2\ell L} \leftarrow \mathcal{MG}_{2\ell L}(1^k)$
$x \leftarrow \mathbb{Z}_p, \tilde{u} \leftarrow \mathbb{G}_\ell, u \leftarrow \mathsf{reRand}_2(\tilde{u})$
$hk_i \leftarrow \mathsf{HGen}(1^k)(i \in [L])$; $msk := x$
$mpk := (\mathcal{MPG}_{2\ell L}, \{hk_i\}_{i\in[L]}, u)$
return (mpk, msk)

Algorithm $\mathsf{Del}(mpk, usk_{\mathbf{id}}, \mathbf{id}')$
parse $\mathbf{id}' =: (id_1, \ldots, id_{d+1})$
if $\mathbf{id} \neq (id_1, \ldots, id_d)$ return $\perp$
$usk_{\mathbf{id}'} \leftarrow \mathsf{reRand}_{d+2}(e(usk_{\mathbf{id}}, \mathsf{H}_{hk_{d+1}}(\mathbf{id}'_{\lceil d+1})))$
return $usk_{\mathbf{id}'}$

Algorithm $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$
$Y_{\mathbf{id}_2} := e(\mathsf{H}_{hk_1}(\mathbf{id}_{2,\lceil 1}), \ldots, \mathsf{H}_{hk_{d_2}}(\mathbf{id}_{2,\lceil d_2}))$
$K_{\mathbf{id}_1,\mathbf{id}_2} := \mathsf{ext}(e(usk_{\mathbf{id}_1}, Y_{\mathbf{id}_2}, \underbrace{u, \ldots, u}_{2L-d_1-d_2 \text{ times}}))$
return $K_{\mathbf{id}_1,\mathbf{id}_2}$

Fig. 2. The H-ID-NIKE scheme $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$.

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 $x \in \mathbb{G}_0$.)

We postpone a proof of the following theorem to Appendix C.

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

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 MPHFs with a random oracle hashing into $\mathbb{G}_1$ in the above scheme $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$ 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 Appendix B.

5 IBE and signature schemes from MPHFs

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: $\mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec}$. Key generation $\mathsf{Gen}(1^k)$, on input a security parameter 1^k, outputs a master public key mpk and a master secret key msk. Key extraction $\mathsf{Ext}(msk, id)$, given msk and an identity $id \in \mathcal{ID}$, outputs a user secret key usk_{id}. Encryption $\mathsf{Enc}(mpk, id, M)$, given mpk, an identity $id \in \mathcal{ID}$, and a message $M \in \mathcal{M}$, outputs a ciphertext C. Decryption $\mathsf{Dec}(usk_{id}, C)$, 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 $(mpk, msk) \leftarrow \mathsf{Gen}(1^k)$, all $id \in \mathcal{ID}$, all $usk_{id} \leftarrow \mathsf{Ext}(msk, id)$, all $M \in \mathcal{M}$, and all $C \leftarrow \mathsf{Enc}(mpk, id, M)$, Dec satisfies $\mathsf{Dec}(usk_{id}, C) = M$.

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 $\mathsf{Ext}(msk, \cdot)$ 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 $C^* \leftarrow \mathsf{Enc}(mpk, id^*, M_b)$ 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 $(\mathsf{poly}, 1)$-MPHFs. Figure 3 depicts $\mathsf{IBE}_{\mathsf{MPHF}}$, which is the Boneh-Franklin IBE scheme [4], implemented with $(\mathsf{poly}, 1)$-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 $(\mathsf{poly}, 1)$-MPHF H into $\mathbb{G}_\ell$. If we take a random oracle as $(\mathsf{poly}, 1)$-MPHF (as in Theorem 1), then $\ell = 1$, and we get the original BF scheme. Correctness of $\mathsf{IBE}_{\mathsf{MPHF}}$ is easy to verify. We now prove its security:

Algorithm $\mathsf{Gen}(1^k)$
$\mathcal{MPG}_{\ell+1} \leftarrow \mathcal{MG}_{\ell+1}(1^k)$
$hk \leftarrow \mathsf{HGen}(1^k), h \leftarrow \mathbb{G}_1, x \leftarrow \mathbb{Z}_p$
$mpk := (\mathcal{MPG}_{\ell+1}, hk, h, \mathsf{reRand}_2(h^x))$
$msk := (hk, x)$
return (mpk, msk)

Algorithm $\mathsf{Ext}(msk, id)$
parse $msk =: (hk, x)$
return $usk_{id} := \mathsf{reRand}_3(\mathsf{H}_{hk}(id)^x)$

Algorithm $\mathsf{Enc}(mpk, id, M)$
parse $mpk =: (\mathcal{MPG}_{\ell+1}, hk, h, \tilde{h})$
$r \leftarrow \mathbb{Z}_p$
$C :=$
$(\mathsf{reRand}_2(h_1^r), \mathsf{ext}(e(\mathsf{H}_{hk}(id), \tilde{h})^r) \oplus M)$
return C

Algorithm $\mathsf{Dec}(usk_{id}, C)$
parse $C =: (C_1, C_2)$
return $M := C_2 \oplus \mathsf{ext}(e(usk_{id}, C_1))$

Fig. 3. The IBE scheme from $(\mathsf{poly}, 1)$-MPHFs.

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

Proof sketch. Assume an IBE-IND-CPA adversary A on $\mathsf{IBE}_{\mathsf{MPHF}}$ that makes q Ext queries. We construct an $(\ell+1)$-MDDH distinguisher B that internally simulates the IBE-IND-CPA experiment for A. B gets as input an $(\ell+1)$-group system $\mathcal{MPG}_{\ell+1}$ and group elements $g, g^{x_1}, \ldots, g^{x_{\ell+2}} \in \mathbb{G}_1$ and $S \in \mathbb{G}_{\ell+1}$, where either $S \stackrel{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{\ell+1}})^{x_{\ell+2}}$ (i.e., S is *real*) or $S \in \mathbb{G}_{\ell+1}$ uniform (i.e., S is *random*). B sets up the master public key as $mpk := (\mathcal{MPG}_{\ell+1}, hk, h, \tilde{h})$ for $(h, \tilde{h}) := (g, g^{x_{\ell+1}})$ and $(hk, td) \leftarrow \mathsf{TGen}(1^k, g^{x_1}, \ldots, g^{x_\ell}, g)$. (Here, we use the $(q,1)$-MPHF property of H, and the corresponding TGen and TEval algorithms.) B can answer an Ext query of A for identity id_i precisely when $a_{id_i} = 0$ (i.e., $\mathsf{TEval}(td, id_i) = (0, B_{id_i})$ for some B_{id_i}): then,

$$usk_{id_i} \leftarrow \mathsf{reRand}_3(e(B_{id_i}, g^{x_{\ell+1}})) \stackrel{\mathsf{grp}}{=} e(B_{id_i}, h)^x \stackrel{\mathsf{grp}}{=} \mathsf{H}_{hk}(id_i)^x.$$

Conversly, we hope that $a_{id^*} \neq 0$ for A's selected challenge identity id^*. Then, we can embed our $(\ell+1)$-MDDH challenge $S \in \mathbb{G}_{\ell+1}$ as

$$C^* \leftarrow (\mathsf{reRand}_2(g^{x_{\ell+2}}), \mathsf{ext}(S^{a_{id^*}} \cdot e(B_{id^*}, g^{x_{\ell+1}}, g^{x_{\ell+2}})) \oplus M_b)$$

for $b \leftarrow \{0,1\}$. Note that if $S \stackrel{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{\ell+1}})^{x_{l+2}}$, this is a valid encryption of M_b; otherwise, C^* contains no information about b. If the simulation has to abort (because $a_{id_i} \neq 0$ for some i, or $a_{id^*} = 0$), then B outputs 0. Otherwise, B outputs 1 iff A correctly guesses b.

If the event that B aborts is independent of the id_i and id^* (as is the case for the RO-based MPHF from Theorem 1), $\Pr[B = 1 \mid S \text{ real}] = \Pr[\neg\mathsf{abort}] \cdot \varepsilon_A$ and $\Pr[B = 1 \mid S \text{ random}] = \Pr[\neg\mathsf{abort}] \cdot 1/2$, where $1/2 + \varepsilon_A$ is the probability that A succeeds in the original IBE-IND-CPA experiment. Hence, B breaks $(\ell+1)$-MDDH iff A breaks $\mathsf{IBE}_{\mathsf{MPHF}}$. However, as in the proof of Theorem 5, we will also have to deal with the case that abort is not independent of the id_i and id^*. An analysis using an "artificial abort" step is necessary that enforces an abort probability that is (almost) independent of the id_i and id^*. The details are as in the proof of Theorem 5. □

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. We postpone a more detailed exposition to Appendix A.

(Hierarchical) signatures from $(\mathsf{poly}, 1)$-MPHFs. 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 $\mathsf{IBE}_{\mathsf{MPHF}}$ 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] Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. From selective to full security: Semi-generic transformations in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 316–333. Springer, May 2012.

[2] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly secure key distribution for dynamic conferences. *Inf. Comput.*, 146(1):1–23, 1998.

[3] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, August 2004.

[4] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, August 2001.

[5] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, December 2001.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[8] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, May 2010.

[9] Çağatay Çapar, Dennis Goeckel, Kenneth G. Paterson, Elizabeth A. Quaglia, Don Towsley, and Murtaza Zafer. Signal-flow-based analysis of wireless security protocols. *Information and Computation*, 226:37–56, 2013.

[10] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 218–227. Springer, November 2007.

[11] Régis Dupont and Andreas Enge. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics*, 154(2):270–276, 2006.

[12] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In *Public Key Cryptography*, pages 254–271, 2013.

[13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, pages 1–17, 2013. Full version under `http://eprint.iacr.org/2012/610`.

[14] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen. Strongly-resilient and non-interactive hierarchical key-agreement in MANETs. In Sushil Jajodia and Javier López, editors, *ESORICS 2008*, volume 5283 of *LNCS*, pages 49–65. Springer, October 2008.

[15] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, December 2002.

[16] Hua Guo, Yi Mu, Zhoujun Li, and Xiyong Zhang. An efficient and non-interactive hierarchical key agreement protocol. *Computers & Security*, 30 (1):28–34, 2011.

[17] Goichiro Hanaoka, Takahiro Matsuda, and Jacob C. N. Schuldt. On the impossibility of constructing efficient key encapsulation and programmable hash functions in prime order groups. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 812–831. Springer, August 2012.

[18] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, August 2008.

[19] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, August 2002.

[20] Kenneth G. Paterson and Sriramkrishnan Srinivasan. 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] M. Ramkumar, N. Memon, and R. Simha. A hierarchical key predistribution scheme. In *Electro Information Technology, 2005 IEEE International Conference*, May 2005. doi: 10.1109/EIT.2005.1626994.

[22] Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In *ISA*, pages 750–759, 2009.

[23] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[24] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

A Hierarchical identity-based encryption from MPHFs

Hierarchical identity-based encryption. A hierarchical identity-based encryption (HIBE) scheme HIBE of depth $D \in \mathbb{N}$ with identity space $\mathcal{ID}$ and message space $\mathcal{M}$ consists of four PPT algorithms. (For a hierarchical user identity we use the vector notation $id = (id_1, \ldots, id_d) \in \mathcal{ID}^d$, with $d \in [D]$. In addition, to denote prefixes of an identity we write $id_{\lceil i} := (id_1, \ldots, id_i)$, for $i \in [d]$.) The key generation $\mathsf{Gen}(1^k, D)$, given the security parameter 1^k in unary and the depth $D \in \mathbb{N}$, outputs a master public key mpk and a master secret key msk. (Note that msk can be seen as the user secret key usk_ε for the empty identity ε.) The key delegation $\mathsf{Del}(mpk, usk_{id}, id')$, given the master public key mpk and the user secret key usk_{id} for an identity $id \in \mathcal{ID}^d$ that is a prefix of

a given $id' \in \mathcal{ID}^{d+1}$, for any $d \in [D-1]$, outputs a user secret key $usk_{id'}$ for identity id'. Given the master public key mpk, an identity $id \in \mathcal{ID}^{\leq D}$, and a message $M \in \mathcal{M}$, $\mathsf{Enc}(mpk, id, M)$ outputs a ciphertext C. The decryption algorithm $\mathsf{Dec}(usk_{id}, C)$, given the user secret key usk_{id} for an identity $id \in \mathcal{ID}^{\leq D}$, and a ciphertext C, outputs a message $M \in \mathcal{M} \cup \{\perp\}$. For correctness, we require for any $k, D \in \mathbb{N}$, all $(mpk, msk) \leftarrow \mathsf{Gen}(1^k, D)$, all $id, id' \in \mathcal{ID}^{\leq D}$ such that id is a prefix of id', all $usk_{id'} \leftarrow \mathsf{Del}(mpk, usk_{id}, id')$, all $M \in \mathcal{M}$, and all $C \leftarrow \mathsf{Enc}(mpk, id', M)$ that $\mathsf{Dec}(usk_{id'}, C) = M$.

HIBE-IND-CPA security. For $D \in \mathbb{N}$ a HIBE scheme HIBE as above is HIBE-IND-CPA-secure iff in the following experiment the success probability of every PPT adversary A is at most negligibly larger than $1/2$. First, the experiment runs $\mathsf{Gen}(1^k, D)$ to obtain a master public mpk and a master secret key msk. Then, A receives mpk and outputs a challenge identity id^* and messages M_0, M_1 of equal length. During the whole experiment, A can query user secret keys for identities of its choice. The experiment answers these queries by iterating Del. (Note that from msk the experiment can derive any user secret key in the hierarchy.) After handing out a prepared ciphertext $C^* \leftarrow \mathsf{Enc}(mpk, id^*, M_b)$ with uniformly chosen $b \leftarrow \{0,1\}$ to A, the adversary outputs a guess b'. If $b = b'$ and A never queried a user secret key for id^* the adversary succeeds.

Extension to HIBE. We can extend our IBE scheme $\mathsf{IBE}_{\mathsf{MPHF}}$ to a HIBE of constant depth D. We work in a $(D\ell+1)$-group system $\mathcal{MPG}_{D\ell+1}$, and use D instances of a $(\mathsf{poly}, 1)$-MPHF H into $\mathbb{G}_\ell$. The resulting HIBE scheme $\mathsf{HIBE}_{\mathsf{MPHF}} = (\mathsf{Gen}, \mathsf{Del}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is given in Figure 4. For simplicity, we assume an identity space $\mathcal{ID}$ such that all involved (potentially multi-level) identities can be uniquely encoded as k-bitstrings. (Larger identities can be used if any identity is first hashed using a collision-resistant hash function.) As in our H-ID-NIKE scheme, we have $msk = usk_\varepsilon = x \in \mathbb{Z}_p = \mathbb{G}_0$; so from msk we can delegate level-1 user secret keys using Del. (In case of an RO as $(\mathsf{poly}, 1)$-MPHF, we get an efficient and simple HIBE in the ROM.) Again, the scheme's correctness is easy to verify. We now turn to the security theorem of $\mathsf{HIBE}_{\mathsf{MPHF}}$.

Theorem 8. *For a constant $D \in \mathbb{N}$ assume a $(D\ell+1)$-group system, and let H be a $(\mathsf{poly}, 1)$-MPHF into $\mathbb{G}_\ell$. Then, the HIBE scheme $\mathsf{HIBE}_{\mathsf{MPHF}} = (\mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ in Figure 4 is HIBE-IND-CPA-secure under the $(D\ell+1)$-MDDH assumption.*

Proof sketch. This system can be proven similar to the IBE case in Theorem 7. Assume a HIBE-IND-CPA adversary A on $\mathsf{HIBE}_{\mathsf{MPHF}}$ with q user secret key queries. We then build a $(D\ell+1)$-MDDH distinguisher B that simulates the HIBE-IND-CPA experiment for A. Given a $(D\ell+1)$-group system $\mathcal{MPG}_{D\ell+1}$, and the challenge $g, g^{x_1}, \ldots, g^{x_{D\ell+2}} \in \mathbb{G}_1$ and $S \in \mathbb{G}_{D\ell+1}$, where either $S \overset{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{D\ell+1}})^{x_{D\ell+2}}$ or $S \in \mathbb{G}_{D\ell+1}$ uniform, B first guesses the length $d^* \in [D]$ of the designated challenge identity $id^* = (id_1, \ldots, id_{d^*})$ that A will output. Then, B sets up the master public key $mpk := (\mathcal{MPG}_{D\ell+1}, hk_1, \ldots, hk_D, h, \tilde{h}, u_2, \ldots, u_D)$ with $(h, \tilde{h}) := (g, g^{x_{D\ell+1}})$ and $(hk_i, td_i) \leftarrow \mathsf{TGen}(1^k, g^{x_{(i-1)\ell+1}}, \ldots,$

Algorithm $\mathsf{Gen}(1^k, D)$
$\mathcal{MPG}_{D\ell+1} \leftarrow \mathcal{MG}_{D\ell+1}(1^k)$
$hk_i \leftarrow \mathsf{HGen}(1^k)$ for all $i \in [D], h \leftarrow \mathbb{G}_1, x \leftarrow \mathbb{Z}_p$
$\tilde{u}_i \leftarrow \mathbb{G}_\ell, u_i \leftarrow \mathsf{reRand}_2(\tilde{u}_i)$ for all $i \in [D], i \neq 1$
$mpk := (\mathcal{MPG}_{D\ell+1}, hk_1, \ldots, hk_D, h, \mathsf{reRand}_2(h^x), u_2, \ldots, u_D)$
$msk := x$
return (mpk, msk)

Algorithm $\mathsf{Del}(mpk, usk_{id}, id')$
parse $mpk =: (\mathcal{MPG}_{D\ell+1}, hk_1, \ldots, hk_D, h, \tilde{h}, u_2, \ldots, u_D)$
parse $id' =: (id_1, \ldots, id_{d+1})$
if $id \neq (id_1, \ldots, id_d)$ return $\perp$
return $\mathsf{reRand}_{d+2}(e(usk_{id}, \mathsf{H}_{hk_{d+1}}(id_{\lceil (d+1) \rceil})))$

Algorithm $\mathsf{Enc}(mpk, id, M)$
parse $mpk =: (\mathcal{MPG}_{D\ell+1}, hk_1, \ldots, hk_D, h, \tilde{h}, u_2, \ldots, u_D)$
parse $id =: (id_1, \ldots, id_d)$, for $d \in [D]$, $r \leftarrow \mathbb{Z}_p$
return $(\mathsf{reRand}_2(h^r), \mathsf{ext}(e(\mathsf{H}_{hk_1}(id_{\lceil 1}), \ldots, \mathsf{H}_{hk_d}(id_{\lceil d}), u_{d+1}, \ldots, u_D, \tilde{h}^r)) \oplus M)$

Algorithm $\mathsf{Dec}(usk_{id}, C)$
parse $C =: (C_1, C_2)$
return $C_2 \oplus \mathsf{ext}(e(usk_{id}, u_{d+1}, \ldots, u_D, C_1))$

Fig. 4. The HIBE scheme from $(\mathsf{poly}, 1)$-MPHFs.

$g^{x_{i\ell}}, g)$, for $i \in [D]$, and $u_i \leftarrow \mathsf{reRand}_2(\tilde{u_i})$ for uniform $\tilde{u_i} \in \mathbb{G}_\ell$, for all $i \leq d^*$. For u_j, with $d^* < j \leq D$, B sets $u_j := \mathsf{reRand}_2(e(g^{x_{(j-1)\ell+1}}, \ldots, g^{x_{j\ell}}))$. Implicitly, we have $msk := x_{D\ell+1}$. To answer user secret key queries for $id = (id_1, \ldots, id_d)$ we hope for $a_{id} := \prod_{i\in[d]} a_{id_{\lceil i}} = 0$, where $(a_{id_{\lceil i}}, B_{id_{\lceil i}}) := \mathsf{TEval}(td_i, id_{\lceil i})$. Thus, we have $a_{id_{\lceil i^*}} = 0$ for some i^* and we can compute the user secret key

$$usk_{id} \leftarrow \mathsf{reRand}_{d+2}(e(\mathsf{H}_{hk_1}(id_{\lceil 1}), \ldots, \mathsf{H}_{hk_{i^*-1}}(id_{\lceil (i^*-1)}), B_{id_{\lceil i^*}}, g^{x_{D\ell+1}}, \mathsf{H}_{hk_{i^*+1}}(id_{\lceil (i^*+1)}), \ldots, \mathsf{H}_{hk_d}(id_{\lceil d}))) \stackrel{\mathsf{grp}}{=} e(\mathsf{H}_{hk_1}(id_{\lceil 1}), \ldots, \mathsf{H}_{hk_d}(id_{\lceil d}))^{msk}.$$

Conversely, we hope that $a_{id^*} := \prod_{i\in[d^*]} a_{id^*_{\lceil i}} \neq 0$ with $(a_{id^*_{\lceil i}}, B_{id^*_{\rceil i}}) = \mathsf{TEval}(td_i, id^*_{\lceil i})$ for the challenge identity $id^* = (id^*_1, \ldots, id^*_{d^*})$. Then, we can embed our $(D\ell+1)$-MDDH challenge $S \in \mathbb{G}_{D\ell+1}$ into the challenge ciphertext $C^* := (C_1, C_2)$ for adversarially chosen M_0, M_1 as follows: set $C_1 \leftarrow \mathsf{reRand}_2(g^{x_{D\ell+2}})$; for C_2 consider the group element

$$e(\mathsf{H}_{hk_1}(id^*_{\lceil 1}), \ldots, \mathsf{H}_{hk_{d^*}}(id^*_{\lceil d^*}), u_{d^*+1}, \ldots, u_D, \tilde{h}^r), \tag{6}$$

whose bit representation (extracted via ext) is used to blind the challenge message M_b.

We can embed B's own challenge S into (6) since (6) contains an implicit $e(g^{x_1}, \ldots, g^{x_{D\ell+1}})^{x_{D\ell+2}\cdot a_{id^*}}$-factor in (6), which can be replaced by $S^{a_{id^*}}$. The remaining $2^{d^*} - 1$ factors of (6) can be computed as during user secret key extraction.

As an example, assume decompositions $\mathsf{H}_{hk_i}(id^*_{\lceil i}) \stackrel{\mathsf{grp}}{=} e(g^{x_{(i-1)\ell+1}}, \ldots, g^{x_{i\ell}})^{a_{id^*_{\lceil i}}} \cdot e(B_{id^*_{\lceil i}}, g)$, and further $u_j \stackrel{\mathsf{grp}}{=} e(g^{x_{(j-1)\ell+1}}, \ldots, g^{x_{j\ell}})$ for $i \leq d^* < j \leq D$, with $D = 3, d^* = 2$. Then

$$\begin{aligned}
e(\mathsf{H}_{hk_1}(id^*_{\lceil 1}), \mathsf{H}_{hk_2}(id^*_{\lceil 2}), u_3, \tilde{h}^r) &\stackrel{\mathsf{grp}}{=} e(e(g^{x_1}, \ldots, g^{x_\ell})^{a_{id^*_{\lceil 1}}} \cdot e(B_{id^*_{\lceil 1}}, g), \\
&\quad e(g^{x_{\ell+1}}, \ldots, g^{x_{2\ell}})^{a_{id^*_{\lceil 2}}} \cdot e(B_{id^*_{\lceil 2}}, g), \\
&\quad e(g^{x_{2\ell+1}}, \ldots, g^{x_{3\ell}}), (g^{x_{3\ell+1}})^{x_{3\ell+2}}) \\
&\stackrel{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{3\ell+1}})^{x_{3\ell+2}\cdot a_{id^*}} \cdot \\
&\quad e(B_{id^*_{\lceil 1}}, g, e(g^{x_{\ell+1}}, \ldots, g^{x_{2\ell}})^{a_{id^*_{\lceil 2}}} \cdot \\
&\quad e(B_{id^*_{\lceil 2}}, g), g^{x_{2\ell+1}}, \ldots, g^{x_{3\ell}}, (g^{x_{3\ell+1}})^{x_{3\ell+2}}) \cdot \\
&\quad e(e(g^{x_1}, \ldots, g^{x_\ell})^{a_{id^*_{\lceil 1}}}, B_{id^*_{\lceil 1}}, g, \\
&\quad g^{x_{2\ell+1}}, \ldots, g^{x_{3\ell}}, (g^{x_{3\ell+1}})^{x_{3\ell+2}}) \\
&\stackrel{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{3\ell+1}})^{x_{3\ell+2}\cdot a_{id^*}} \cdot \\
&\quad e(B_{id^*_{\lceil 1}}, g^{x_{3\ell+2}}, e(g^{x_{\ell+1}}, \ldots, g^{x_{2\ell}})^{a_{id^*_{\lceil 1}}} \cdot \\
&\quad e(B_{id^*_{\lceil 2}}, g), g^{x_{2\ell+1}}, \ldots, g^{x_{3\ell}}, g^{x_{3\ell+1}}) \cdot \\
&\quad e(e(g^{x_1}, \ldots, g^{x_\ell})^{a_{id^*_{\lceil 1}}}, B_{id^*_{\lceil 1}}, \\
\end{aligned}$$

$$g^{x_{3\ell+2}}, g^{x_{2\ell+2}}, \ldots, g^{x_{3\ell}}, g^{x_{3\ell+1}}).$$

Now, back to the general case, C^* is a valid encryption of M_b iff $S \stackrel{\mathsf{grp}}{=} e(g^{x_1}, \ldots, g^{x_{D\ell+1}})^{x_{D\ell+2}}$. Else, for uniform $S \in \mathbb{G}_{D\ell+1}$, the ciphertext C^* contains no information about b. Further, if B has to abort because of $a_{id_i} \neq 0$ for some i or $a_{id^*} = 0$ for the challenge identity id^*, then B ouputs 0. Otherwise, if the adversary guesses b correctly, B ouputs 1. Again, the event that B aborts might not be independent of id_i for some i and id^*. Hence, we have to implement an artificial abort step such that the probability of aborting B's simulation is (almost) independent of the id_i and id^*. This is done as in the proof of Theorem 5. □

B (H-)ID-NIKE with multiple TAs and group-H-ID-NIKE

Multilinear maps are sufficiently powerful to allow further, powerful generalisations of our (H)-ID-NIKE construction. In particular, we may consider a situation where we have multiple TAs, each issuing secret keys to (a hierarchy of) users, and we wish to enable any pair of users with secret keys issued by possibly different TAs to be able to compute a shared key. Going further, we may wish to enable groups of users (rather than just pairs of users) in the "forest" of hierarchies to compute shared keys. All of this is enabled in the multilinear setting by generalisation of our ID-NIKE and H-ID-NIKE schemes. For simplicity, we sketch just one such scheme here, leaving detailed development of these ideas to future work.

Suppose we have a 3-group system and let H be a random oracle with outputs in $\mathbb{G}_1$. Then we can instantiate our ID-NIKE scheme with the MPHF being replaced by H and with secret keys of the form $usk_{id,i} \leftarrow \mathsf{reRand}_2(\mathsf{H}(id)^{msk_i}) \in \mathbb{G}_1$. Now assume we have two trusted authorities TA_1, TA_2 with master secrets msk_1, msk_2. We augment mpk_i to include the element $h_i = \mathsf{reRand}_2(g^{msk_i})$. Consider the chain of equalities:

$$\begin{aligned} e(usk_{id_1,1}, \mathsf{H}(id_2), h_2) &\stackrel{\mathsf{grp}}{=} e(\mathsf{H}(id_1)^{msk_1}, \mathsf{H}(id_2), g^{msk_2}) \\ &\stackrel{\mathsf{grp}}{=} \ldots \stackrel{\mathsf{grp}}{=} e(usk_{id_2,2}, \mathsf{H}(id_1), h_1). \end{aligned}$$

The first computation in the chain can be carried out by user id_1 using its secret key issued by TA_1, while the last can be done by id_2 using its secret key issued by TA_2; thus the pairing output can be used as the basis of a shared key (by applying ext in the usual way). Hence two users with secret keys issued by *different* TAs can still compute a shared key non-interactively. It should be evident how to generalise this simple scheme a) to the standard model, b) to greater numbers of users and TAs, and c) to the hierarchical setting, all by working with ℓ-group systems.

C Proof of Theorem 6

Proof. The proof is very similar to the proof of Theorem 5; we focus on the necessary adaptations. We construct a $(2\ell L+1)$-power distinguisher from an IND-SK adversary A. Assume B gets a $2\ell L$-group system $\mathcal{MPG}_{2\ell L}$ and group elements $g, g^x \in \mathbb{G}_1$ and $S \in \mathbb{G}_{2\ell L}$ as input, and is supposed to distinguish the cases $S \stackrel{\mathsf{grp}}{=} e(g^x,\ldots,g^x)^x$ and random S.

B simulates the IND-SK experiment for A. First, B runs $(hk_i, td_i) \leftarrow \mathsf{TGen}(1^k, g^x,\ldots,g^x,g)$ and for $u \leftarrow \mathsf{reRand}_2(e(\underbrace{g^x,\ldots,g^x}_{\ell \text{ times}}))$, it sets $mpk := (\mathcal{MPG}_{2\ell L}, \{hk_i\}_{i\in[L]}, u)$. To answer an ***Extract*** query for identity $\mathbf{id} = (id_1,\ldots,id_d)$, B will hope for $a_{\mathbf{id}} := \prod_{i\in[d]} a_{i,\mathbf{id}_{\lceil i}} = 0$, where $(a_{i,\mathbf{id}_{\lceil i}}, B_{i,\mathbf{id}_{\lceil i}}) := \mathsf{TEval}(td_i, \mathbf{id}_{\lceil i})$. In that case, we must have $a_{i^*,\mathbf{id}_{\lceil i^*}} = 0$ for some i^*, and thus B can compute $usk_{\mathbf{id}}$ using

$$usk_{\mathbf{id}} \leftarrow \mathsf{reRand}_{d+2}(e(\mathsf{H}_{hk_1}(\mathbf{id}_{\lceil 1}),\ldots,\mathsf{H}_{hk_{i^*-1}}(\mathbf{id}_{\lceil i^*-1}), e(B_{\mathbf{id}_{\lceil i^*}}, g^x), \mathsf{H}_{hk_{i^*+1}}(\mathbf{id}_{\lceil i^*+1}),\ldots,\mathsf{H}_{hk_d}(\mathbf{id}_{\lceil d}))) \stackrel{\mathsf{grp}}{=} e(\mathsf{H}_{hk_1}(\mathbf{id}_{\lceil 1}),\ldots,\mathsf{H}_{hk_d}(\mathbf{id}_{\lceil d}))^{msk}.$$

Conversely, B can embed its own challenge S into the challenge key K^* whenever the challenge identities $\mathbf{id}_1^* = (id_{1,1}^*,\ldots,id_{1,d_1^*}^*)$ and $\mathbf{id}_2^* = (id_{2,1}^*,\ldots,id_{2,d_2^*}^*)$ satisfy $a_{\mathbf{id}_1^*}, a_{\mathbf{id}_2^*} \neq 0$. Namely, in that case, the group element

$$e(\mathsf{H}_{hk_1}(\mathbf{id}_{1,\lceil 1}^*),\ldots,\mathsf{H}_{hk_{d_1^*}}(\mathbf{id}_{1,\lceil d_1^*}^*), \mathsf{H}_{hk_1}(\mathbf{id}_{2,\lceil 1}^*),\ldots,\mathsf{H}_{hk_{d_2^*}}(\mathbf{id}_{2,\lceil d_2^*}^*), \underbrace{u,\ldots,u}_{2L-d_1^*-d_2^* \text{ times}})^{msk}, \tag{7}$$

from which the shared key is computed, contains an $e(\overbrace{g^x,\ldots,g^x}^{2\ell L \text{ times}})^x$-factor, which can be replaced by B's own challenge S; the remaining $2^{d_1+d_2}-1$ factors of (7) can be computed as in (5).

Hence, B's simulation requires that $a_{\mathbf{id}} = 0$ for all non-challenge queried identities $\mathbf{id}$, and $a_{\mathbf{id}_1^*}, a_{\mathbf{id}_2^*} \neq 0$. It will be sufficient to hope for $a_{\mathbf{id}_{j,\lceil i}^*} \neq 0$ for all prefixes of the challenge identities, and $a_{\mathbf{id}_{\lceil i}} = 0$ for all other involved prefixes. (Since no prefixes of the challenge identities will need to be extracted, these requirements are not contradictory.) These requirements translate to requirements on the L individual MPHF instances. Hence, with probability at least $(1/p(k))^L$ (for the polynomial $p(k)$ from (2)), the simulation will not abort.

The remaining analysis (including a necessary artificial abort step) can be performed as in the proof of Theorem 5. □