Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures

Susan Hohenberger[1,⋆], Amit Sahai [⋆⋆], and Brent Waters [⋆⋆⋆]

[1] Johns Hopkins University, `susan@cs.jhu.edu`
[2] UCLA, `sahai@cs.ucla.edu`
[3] University of Texas at Austin, `bwaters@cs.utexas.edu`

Abstract. In this work, we explore building constructions with full domain hash structure, but with standard model proofs that do not employ the random oracle heuristic. The launching point for our results will be the utilization of a "leveled" *multilinear map* setting for which Garg, Gentry, and Halevi (GGH) recently gave an approximate candidate. Our first step is the creation of a standard model signature scheme that exhibits the structure of the Boneh, Lynn and Shacham signatures. In particular, this gives us a signature that admits unrestricted aggregation.
We build on this result to offer the first *identity-based* aggregate signature scheme that admits unrestricted aggregation. In our construction, an arbitrary-sized set of signatures on identity/message pairs can be aggregated into a single group element, which authenticates the entire set. The identity-based setting has important advantages over regular aggregate signatures in that it eliminates the considerable burden of having to store, retrieve or verify a set of verification keys, and minimizes the total cryptographic overhead that must be attached to a set of signer/message pairs. While identity-based signatures are trivial to achieve, their aggregate counterparts are not. To the best of our knowledge, no prior candidate for realizing unrestricted identity-based aggregate signatures exists in either the standard or random oracle models.

[⋆] Supported by the National Science Foundation (NSF) CNS-1154035, CNS-1228443; the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211, the Office of Naval Research under contract N00014-11-1-0470, and a Microsoft Faculty Fellowship.

[⋆⋆] Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389.

[⋆⋆⋆] Supported by NSF CNS-0915361, CNS-0952692, CNS-1228599; DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, a Google Faculty Research Award, an Alfred P. Sloan Fellowship, a Microsoft Faculty Fellowship, and a Packard Foundation Fellowship. Applying to all authors, the views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

A key technical idea underlying these results is the realization of a hash function with a Naor-Reingold-type structure that is publicly computable using repeated application of the multilinear map. We present our results in a generic "leveled" multilinear map setting and then show how they can be translated to the GGH graded algebras analogue of multilinear maps.

1 Introduction

Applying a full domain hash is a common technique in cryptography where a hash function, modeled as a random oracle, is used to hash a string into a set. Originally, the concept referred to a signature scheme where one hashed into the range of a trapdoor permutation [3]. Subsequently, full domain hash has been treated as a more general concept and applied in bilinear map cryptography where typically a hash function $H : \{0,1\}^* \to \mathbb{G}$ is used to hash a string into a bilinear group. (We note that multiple early works [9, 11, 10] employ this terminology.) Pairing-based applications of Full Domain Hash include: the original Boneh-Franklin [9], short and aggregate signatures [11, 10], Hierarchical Identity-Based Encryption [23], and decentralized Attribute-Based Encryption [26]. Typically, proofs of such schemes will use the random oracle heuristic to "program" the output of the hash function in a certain way for which there is no known standard model equivalent (see [24]).

Given that there are well-known issues with random oracle instantiability in general [14] and problems with Full Domain Hash in particular [18, 17], there has been a push to find standard model realizations of these applications. These endeavors have been successful in several applications such as signatures [8, 36] and (Hierarchical) Identity-Based Encryption [15, 6, 7, 36, 21, 37]. Despite this progress, the current state is not entirely satisfactory on two fronts. First, each of the standard model examples given above created new cryptographic constructions with fundamentally different structure than the original Full Domain Hash construction. While creating a new structure is a completely valid and novel approach, that path does not necessarily lend insight or further understanding of the original constructions.

Second, there are important applications of the Full Domain Hash method where implementing such a hash using a random oracle introduces significant limitations in the applicability of the Full Domain Hash method. One example concerns aggregate signature schemes and their identity-based counterparts.

An aggregate signature system is one in which a signature σ' on verification key/message pair (VK', M') can be combined with a signature $\tilde{\sigma}$ on $(\tilde{\mathrm{VK}}, \tilde{M})$ producing a new signature σ on the set $S = \{(\mathrm{VK}', M'), (\tilde{\mathrm{VK}}, \tilde{M})\}$. This process can be repeated indefinitely to aggregrate an arbitrary number of signatures together. Crucially, the size of σ should be independent of the number of signatures aggregated, although the description of the set S will grow. The ultimate goal, however, is to minimize the entire transmission size [31].

The need for a public-key infrastructure for verification keys is a major drawback of traditional public-key cryptography, and for this reason identity-based cryptography has flourished [35, 9]: In an *identity-based* aggregate signature scheme, verification keys like VK would be replaced with simple identity strings like $\mathcal{I} =$"harrypotter@hogwarts.edu". This offers a very meaningful savings for protocols such as BGPsec, which require routers to store, retrieve and verify certificates for over 36,000 public keys [16, 13]. We note that while identity-based signatures follow trivially from standard signatures, identity-based aggregate signatures are nontrivial (more on this below).

A decade ago, the Boneh, Gentry, Lynn and Shacham (BGLS) [10] aggregate signature scheme was built using the Full Domain Hash methodology. In the original vision of BGLS, aggregation could be performed by any third party on any number of signatures. The authors showed how the Boneh, Lynn and Shacham (BLS) [11] signatures (which are in turn comprised of Boneh-Franklin [9] private IBE keys) can be aggregated in this manner. The BLS construction uses a full domain hash and its security proof is in the random oracle model. However, even though the BGLS scheme was built upon the key mechanism for Boneh-Franklin Identity-Based Encryption, BGLS does *not* support identity-based aggregation. The Full Domain Hash in BGLS is realized using a random oracle, which destroys the structure that would be needed for identity-based aggregate signatures. To the best of our knowledge, no prior solution to identity-based aggregate signatures in either the standard or random oracle models exists. Prior work considered ID-based aggregates restricted to a common nonce [22] (e.g., where signatures can only be aggregated if they were created with the same nonce or time period) or sequential additions [5] (e.g., where a group of signers sequentially form an aggregate by each adding their own signature to the aggregate-so-far).

Our results in a nutshell. In this work, we give a new method for implementing the Full Domain Hash method using leveled multilinear maps, including the ones recently proposed by Garg, Gentry, and Halevi (GGH) [19]. We show how to use this method to implement aggregate signatures in the standard model in a way that naturally extends to give the first full solution to the problem of identity-based aggregate signatures (also in the standard model).

Prior work on standard model aggregate signatures. All previous work on achieving standard model aggregate signatures did so by departing fundamentally from the Full Domain Hash methodology.

Subsequently to BGLS [10], different standard model solutions were proposed, but with different restrictions on aggregation. These include: constructions [27] where the signatures must be sequentially added in by the signers, multisignatures [27] where aggregation can occur only for the same message M, or where aggregation is limited to signatures associated with the same nonce or time period [1].[4] These restrictions limit their practical applicability.

[4] We remark that these restrictions were considered in other works such as [33, 32, 29, 4, 28] prior to the standard model constructions cited above.

In 2009, Rückert and Schröder [34] gave an intriguing vision on how multilinear maps might enable standard model constructions of aggregate signatures, also departing from the Full Domain Hash methodology. They did not discuss or achieve identity-based aggregate signatures. Their proposal came before the Garg, Gentry and Halevi [19] candidate and used the earlier Boneh-Silverberg [12] view of multilinear maps, where a k-linear map would allow the *simultaneous* multiplication of k source group elements into one target group element. The GGH candidate in contrast allows for encodings to exist on multiple levels and a pairing between an encoding on level i and one on level j gives an encoding on level $i+j$ as long as $i+j$ is less than or equal to some k. One drawback of the Rückert and Schröder construction is that the security proof requires access to an interactive (or oracle-type) assumption in order to answer the signature queries where the structure of the oracle output is essentially identical to the signatures required. This property seems to be tightly coupled with the modeling of a multilinear map as a one time multiplication. In contrast, we will exploit the leveling of the GGH abstraction to actually replace the hash function in a BLS-type structure and obtain proofs from non-interactive assumptions.

1.1 Overview of our Aggregate Signature Constructions

We now overview the constructions and their security claims. To simplify the description of the main ideas, we describe the constructions here in terms of leveled multilinear maps. Later on, we give translations to the GGH framework.

The Base Construction. A trusted setup algorithm will take as input security parameter λ and message bit-length ℓ and run a group generator $\mathcal{G}(1^{\lambda}, k=\ell+1)$ and outputs a sequence of groups $\mathbb{G}=(\mathbb{G}_1,\ldots,\mathbb{G}_k)$ of prime order p.[5] The group sequence will have canonical generators $g=g_1,g_2,\ldots,g_k$ along with a pairing operation that computes $e(g_i^a, g_j^b)=g_{i+j}^{ab}$ for any $a,b\in\mathbb{Z}_p$ and $i+j\le k$. The setup algorithm will also choose $\boldsymbol{A}=(A_{1,0}=g^{a_{1,0}}, A_{1,1}=g^{a_{1,1}}),\ldots,(A_{\ell,0}=g^{a_{\ell,0}}, A_{\ell,1}=g^{a_{\ell,1}})\in\mathbb{G}_1^2$. We define $H:\{0,1\}^{\ell}\to\mathbb{G}_{k-1}$ as $H(M)=g_{k-1}^{\prod_{i\in[1,\ell]} a_{i,m_i}}$, where m_i are the bits of message M. The hash function hashes a message into the group $\mathbb{G}_{k-1}$. It exhibits a Naor-Reingold [30]-type structure and is publicly computable using repeated application of a multilinear map. Since a group element in $\mathbb{G}_{k-1}$ has one pairing left, it intuitively reflects the bilinear map setting. In our scheme a private key contains a random exponent $\alpha\in\mathbb{Z}_p$ and the corresponding verification key VK contains g^{α}. A signature on a message M is computed as $\sigma=H(M)^{\alpha}$ and verified by testing $e(\sigma,g)\overset{?}{=}e(H(M),g^{\alpha})$.

Stepping back, the structure of our scheme very closely resembles BLS signatures. For this reason it is possible to aggregate them in the BGLS fashion by simply multiplying two together. The size of an aggregate signature depends on the security parameter plus message length ℓ (assuming the group representation size increases with $k=\ell+1$), but is independent of the number of times

[5] In practice one will perform a CRHF of an arbitrary length message to ℓ bits.

aggregation is applied. Aggregation is unrestricted and can be done by any third party.

The Rückert and Schröder construction [34] also insightfully uses a Naor-Reingold type function for aggregation. A key distinction is that in the RS method there is a *unique* NR function for each signer and it is privately computed by each signer per each message/input. In our construction the Naor-Reingold function is computed as a *public* hash using the levels of the multilinear map. A signer simply multiplies in his secret exponent after computing the hash. Thus, this mimicks the BLS structure much more closely. One advantage of our structure is that the hash function can be derived from a single common reference string and then public keys are just a single group element. In addition, we will see that our structure is amenable to proofs under non-interactive assumptions and allow us to extend to the identity-based setting. In the aggregation setting, where bandwidth is at a premium, our smaller public keys and the ability to go identity-based is important.

Proofs of Security. We view our aggregate signatures as signatures on a multi-set of message/verification key pairs for full generality. We prove security in a modular way as a two step process. First, we define a weaker "distinct message" variant of security that only considers an attacker successful if the aggregate forgery no two signers sign the same message. We then show how to transform any distinct message secure scheme into one with standard security. The transformation captures the BGLS idea (formalized by Bellare, Namprempre and Neven [2]) of hashing the public key plus message together. Using the transformation we can focus on designing proofs in the distinct message game. We first prove selective security under a natural analog of the CDH assumption we call the k-Multilinear Computational Diffie-Hellman (k-MCDH) assumption. We next show full (a.k.a., adaptive) security using a subexponentially secure version of the assumption. Finally, we show full security with only polynomial factors in the reduction using a non-interactive, but parameterized assumption.

Realizing Identity-Based Aggregation. The authority will run a setup algorithm that takes the message bit-length ℓ and identity bit-length n. It runs a group generator $\mathcal{G}(1^\lambda, k = \ell + n)$ and outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p. It creates the parameters $\boldsymbol{A}$ as in the prior scheme and $\boldsymbol{B} = (B_{1,0} = g^{b_{1,0}}, B_{1,1} = g^{b_{1,1}}), \ldots, (B_{n,0} = g^{b_{n,0}}, B_{n,1} = g^{b_{n,1}}) \in \mathbb{G}_1^2$. We define $H : \{0,1\}^n \times \{0,1\}^\ell \to \mathbb{G}_{k-1}$ as $H(\mathcal{I}, M) = g_k^{(\prod_{i\in[1,n]} b_{i,\mathrm{id}_i})(\prod_{i\in[1,\ell]} a_{i,m_i})}$, where m_i are the bits of message M and id_i the bits of $\mathcal{I}$. The hash function is publicly computable from the multilinear map. A secret key for identity $\mathcal{I}$ is computed as $\mathrm{SK}_{\mathcal{I}} = g_{n-1}^{\prod_{i\in[1,n]} b_{i,\mathrm{id}_i}} \in G_{n-1}$. This can be used to produce a signature on message M by computing $(g_{k-1})^{(\prod_{i\in[1,n]} b_{i,\mathrm{id}_i})(\prod_{i\in[1,\ell]} a_{i,m_i})}$ using the multilinear map. Finally, a signature can be verified by checking $e(\sigma, g) \stackrel{?}{=} H(\mathcal{I}, M)$. The signatures will aggregate in the same manner by multiplying together.

The distinct message translation is not required in the identity-based setting, because there is no rogue key problem. We first prove selective security under the k-MCDH assumption, and then show full security using a subexponentially secure version of the assumption. We provide these proofs in both the generic multilinear and the GGH framework.

Further Applications. Taken altogether we show that multilinear forms provide an opportunity for revisiting cryptographic structures that were strongly associated with the random oracle heuristic. It remains to be seen how widely this direction will apply. One interesting example of an application that currently requires the full domain hash is the decentralized Attribute-Based Encryption system of Lewko and Waters [26]. There is no standard model candidate that has comparable expressiveness. Here performing an analogous transformation to our aggregate signatures hash function gives a candidate construction that we do not immediately see how to break. However, it is less easy to see how our proof techniques would extend to the variant of the Lewko-Waters [26] decentralized ABE scheme.

2 Leveled Multilinear Maps and the GGH Graded Encoding

We give a description of generic, leveled multilinear maps. The assumptions used in this setting are defined inline with their respective security proofs. Basic details of the GGH graded algebras analogue of mulitlinear maps are included where used, and for further details, please refer to [19].

For generic, leveled multilinear maps, we assume the existence of a group generator $\mathcal{G}$, which takes as input a security parameter 1^λ and a positive integer k to indicate the number of allowed pairing operations. $\mathcal{G}(1^\lambda, k)$ outputs a sequence of groups $\boldsymbol{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ each of large prime order $p > 2^\lambda$. In addition, we let g_i be a canonical generator of $\mathbb{G}_i$ (and is known from the group's description). We let $g = g_1$.

We assume the existence of a set of bilinear maps $\{e_{i,j} : G_i \times G_j \to G_{i+j} \mid i, j \geq 1;\ i + j \leq k\}$. The map $e_{i,j}$ satisfies the following relation:

$$e_{i,j}\left(g_i^a, g_j^b\right) = g_{i+j}^{ab} \ : \ \forall a, b \in \mathbb{Z}_p$$

We observe that one consequence of this is that $e_{i,j}(g_i, g_j) = g_{i+j}$ for each valid i, j.

When the context is obvious, we will sometimes abuse notation and drop the subscripts i, j, For example, we may simply write $e\left(g_i^a, g_j^b\right) = g_{i+j}^{ab}$.

Algorithmic components of GGH encodings. While we assume familiarity with the basics of GGH encodings [19], we now review the algorithmic components of the GGH encodings that we will use in our constructions and proofs. The setup algorithm $\mathsf{InstGen}(1^\lambda, 1^k)$ takes as input a security parameter 1^λ and the level of multilinearity 1^k, and outputs the public parameters params needed for using the remaining GGH algorithms, along with a special parameter $\mathbf{p}_{zt}$ to be used for zero testing. The sampling algorithm $\mathsf{samp}(\mathsf{params})$ outputs a level-0 encoding of a randomly chosen element. The canonicalizing encoding $\mathsf{cenc}_e(\mathsf{params}, i, \alpha)$ algorithm takes as input an encoding α of some element a, and outputs a level-i encoding of a, with re-randomization parameter e. This canonicalizing encoding algorithm can re-randomize an encoding for a fixed constant number of re-randomization parameters e. Finally, the zero-testing algorithm $\mathsf{isZero}(\mathbf{p}_{zt}, \alpha)$ takes as input a level-k encoding α, and accepts iff α is an encoding of 0. A more elaborate review of these algorithms can be found elsewhere in these proceedings [20] (omitted here for lack of space).

3 Definitions for Aggregate and ID-based Aggregate Signatures

We now give our definitions for aggregate signatures. In our setting, each aggregate signature is associated with a *multiset* S over verification key/message pairs (or identity/message pairs in the ID-based setting). A set S is of the form $\{(\mathrm{VK}_1, M_1), \ldots, (\mathrm{VK}_{|S|}, M_{|S|})\}$. Since S is a multiset it is possible to have $(\mathrm{VK}_i, M_i) = (\mathrm{VK}_j, M_j)$ for $i \neq j$. All signatures, including those that come out of the sign algorithm, are considered to be aggregate signatures. The aggregation algorithm is general in that it can take any two aggregate signatures and combine them into a new aggregate signature.

Our definition allows for an initial trusted setup that will generate a set of common public parameters PP. This will define a bit length of all messages (and identities). In practice one could set these fixed lengths to be the output length ℓ of a collision resistant hash function and allow arbitrary-length messages/identities by first hashing them down to ℓ bits. In the ID-based setting, the authority also produces a master secret key used later to run the key generation algorithm.

We emphasize a few features of our setting. First, aggregation is very general in that it allows for the combination of any two aggregate signatures into a single one. Some prior definitions required an aggregate signature to be combined with a single message signature. This is a limitation for applications where an aggregator comes across two aggregate signatures that is wishes to combine. The aggregation operation does not require any secret keys. The multiset structure allows one to combine two aggregate signatures which both include the same message from the same signer.

We begin formally with the ID-based definition, because it is novel to this work, and then discuss its simpler counterpart.

Authority-Setup$(1^\lambda, \ell, n)$ The trusted setup algorithm takes as input the security parameter as well the bit-length ℓ of messages and bit-length n of the identities. It outputs a common set of public parameters PP and master secret key MSK.

KeyGen$(\mathrm{MSK}, \mathcal{I} \in \{0,1\}^n)$ The key generation algorithm is run by the authority. It takes as input the system master secret key and an identity $\mathcal{I}$, and outputs a secret signing key $\mathrm{SK}_{\mathcal{I}}$.

Sign($\mathrm{PP}, \mathrm{SK}_{\mathcal{I}}, \mathcal{I} \in \{0,1\}^n, M \in \{0,1\}^\ell$*)* The signing algorithm takes as input a secret signing key and corresponding identity $\mathcal{I} \in \{0,1\}^n$, the common public parameters as well as a message $M \in \{0,1\}^\ell$. It outputs a signature σ for identity $\mathcal{I}$. *We emphasize that a single signature that is output by this algorithm is considered to also be an aggregate signature.*

Aggregate($\mathrm{PP}, \tilde{S}, S', \tilde{\sigma}, \sigma'$*).* The aggregation algorithm takes as input two multisets $\tilde{S}$ and S' and purported signatures $\tilde{\sigma}$ and σ'. The elements of $\tilde{S}$ consist of identity/message pairs $\{(\tilde{\mathcal{I}}_1, \tilde{M}_1), \ldots, (\tilde{\mathcal{I}}_{|\tilde{S}|}, \tilde{M}_{|\tilde{S}|})\}$ and the elements of S' consist of $\{(\mathcal{I}'_1, M'_1), \ldots, (\mathcal{I}'_{|S'|}, M'_{|S'|})\}$. The process produces a signature σ on the multiset $S = \tilde{S} \cup S'$, where $\cup$ is a multiset union.

Verify(PP, S, σ*).* The verification algorithm takes as input the public parameters, a multiset S of identity and message pairs and an aggregate signature σ. It outputs true or false to indicate whether verification succeeded.

Correctness The correctness property states that all valid aggregate signatures will pass the verification algorithm, where a valid aggregate is defined recursively as an aggregate signature derived by an application of the aggregation algorithm on two valid inputs or the signing algorithm. More formally, for all integers $\lambda, \ell, n, k \geq 1$, all $\mathrm{PP} \in \text{Authority-Setup}(1^\lambda, \ell, n)$, all $\mathcal{I}_1, \ldots, \mathcal{I}_k \in \{0,1\}^n$, all $\mathrm{SK}_{\mathcal{I}_i} \in \mathrm{KeyGen}(\mathrm{PP}, \mathcal{I}_i)$, $\mathrm{Verify}(\mathrm{PP}, S, \sigma) = 1$, if σ is a *valid* aggregate for multiset S under PP. We say that an aggregate signature σ is *valid* for multiset S if: (1) $S = \{(\mathcal{I}_i, M)\}$ for some $i \in [1, k]$, $M \in \{0,1\}^\ell$ and $\sigma \in \mathrm{Sign}(\mathrm{PP}, \mathrm{SK}_{\mathcal{I}_i}, \mathcal{I}_i, M)$; or (2) there exists multisets $S', \tilde{S}$ where $S = S' \cup \tilde{S}$ and valid aggregate signatures $\sigma', \tilde{\sigma}$ on them respectively such that $\sigma \in \mathrm{Aggregate}(\mathrm{PP}, \tilde{S}, S', \tilde{\sigma}, \sigma')$.

Security Model for Aggregate Signatures. Adapting aggregation [10, 2] to the identity-based setting takes some care in considering how keys are handled and which query requests the adversary should be allowed to make. Informally, in the unforgeability game, it should be computationally infeasible for any adversary to produce a forgery implicating an honest identity, even when the adversary can control all other identities involved in the aggregate and can mount a chosen-message attack on the honest identity. This is defined using a game between a challenger and an adversary $\mathcal{A}$ with respect to scheme $\mathit{\Pi}$ = (Authority-Setup, KeyGen, Sign, Aggregate, Verify).

– **ID-Unforg**$(\mathit{\Pi}, \mathcal{A}, \lambda, \ell, n)$:

Setup. The challenger runs Authority-Setup$(1^\lambda, \ell, n)$ to obtain PP. It sends PP to $\mathcal{A}$.

Queries. Proceeding adaptively, $\mathcal{A}$ can make three types of requests:

1. Create New Key: The challenger begins with an index $i = 1$ and an empty sequence of index/identity/private key triples T. On input an identity $\mathcal{I} \in \{0,1\}^n$, the challenger runs KeyGen$(\mathrm{MSK}, \mathcal{I})$ to obtain $\mathrm{SK}_{\mathcal{I}}$. It adds the triple $(i, \mathcal{I}, \mathrm{SK}_{\mathcal{I}})$ to T and then increments i for the next

call. Nothing is returned to the adversary. *We note that the adversary can query this oracle multiple times for the same identity. This will capture security for applications that might release more than one secret key per identity.*

2. Corrupt User: On input an index $i \in [1, |T|]$, the challenger returns to the adversary the triple $(i, \mathcal{I}_i, \mathrm{SK}_{\mathcal{I}_i}) \in T$. It returns an error if T is empty or i is out of range.
3. Sign: On input an index $i \in [1, |T|]$ and a message $M \in \{0,1\}^\ell$, the challenger obtains the triple $(i, \mathcal{I}_i, \mathrm{SK}_{\mathcal{I}_i}) \in T$ (returning an error if it does not exist) and returns the signature resulting from $\mathrm{Sign}(\mathrm{PP}, \mathrm{SK}_{\mathcal{I}_i}, \mathcal{I}_i, M)$ to $\mathcal{A}$.

Response. Finally, $\mathcal{A}$ outputs a multiset S^* of identity/message pairs and a purported aggregate signature σ^*.

We say the adversary "wins" or that the output of this experiment is 1 if: (1) $\mathrm{Verify}(\mathrm{PP}, S^*, \sigma^*) = 1$ and (2) there exists an element $(\mathcal{I}^*, M^*) \in S^*$ such that M^* was not queried for a signature by the adversary on any index corresponding to $\mathcal{I}^*$; i.e., any index i such that $(i, \mathcal{I}^*, \cdot) \in T$. Otherwise, the output is 0. Define $\textbf{ID-Forg}_{\mathcal{A}}$ as the probability that $\textbf{Unforg}(\mathit{\Pi}, \mathcal{A}, \lambda, \ell, n) = 1$, where the probability is over the coin tosses of the Authority-Setup, KeyGen, and Sign algorithms and of $\mathcal{A}$.

Definition 1 (Adaptive Unforgeability). *An ID-based aggregate signature scheme $\mathit{\Pi}$ is existentially* unforgeable *with respect to adaptive chosen-message attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the function **ID-Forg**$_A$ is negligible in λ.*

Selective Security. We consider a selective variant to **ID-Unforg** (selective in both the identity and the message) where there is an Init phase before the Setup phase, wherein $\mathcal{A}$ gives to the challenger a forgery identity/message pair $(\mathcal{I}^* \in \{0,1\}^n, M^* \in \{0,1\}^\ell)$. The adversary cannot request a signing key for $\mathcal{I}^*$. (It may request that the challenger create one or more keys for this identity, but it cannot corrupt any user index i associated with $\mathcal{I}^*$.) Moreover, the adversary only "wins" causing the experiment output to be 1 if the normal checks hold (i.e., its signature verifies and it did not request that $\mathcal{I}^*$ sign M^*) and additionally $(\mathcal{I}^*, M^*)$ appears in S^*.

Non-ID-Based Aggregates and the Distinct Message Variant. We provide security definitions for the non-ID-based setting in the full version [25] that follow from [10, 2]. We provide adaptive and selective variants. We also identify a weaker "distinct message" security game that is easier to work with. In the full version [25], we describe and prove secure a simple transformation from distinct message security to standard aggregate signature security. The transformation captures the idea of hashing the public key and message together [10, 2] in a modular way. Focusing on distinct message security allows one to avoid the "rogue key" attack (see Section 4.2). We do not consider distinct message security in the ID-based setting, because there are no verification keys.

4 Our Base Aggregate Signature Construction

4.1 Generic Multlinear Construction

Setup($1^\lambda, \ell$*)* The trusted setup algorithm takes as input the security parameter as well as the length ℓ of messages. It first runs $\mathcal{G}(1^\lambda, k = \ell + 1)$ and outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$, where we let $g = g_1$.

Next, it outputs random group elements $(A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell,1}) \in \mathbb{G}_1^2$. These will be used to compute a function $H(M) : \{0,1\}^\ell \to \mathbb{G}_{k-1}$, which serves as the analog of the full domain hash function of the BGLS [10] construction. Let $m_1, \ldots, m_\ell$ be the bits of message M. It is computed iteratively as $H_1(M) = A_{1,m_1}$ and for $i \in [2, \ell]$, $H_i(M) = e(H_{i-1}(M), A_{i,m_i})$. We define $H(M) = H_\ell(M)$. The public parameters, PP, consist of the group descriptions plus $(A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell,1})$.

*KeyGen(*PP*)* The key generation algorithm first chooses random $\alpha \in \mathbb{Z}_p$. It outputs the public verification key as $\mathrm{VK} = g^\alpha$. The secret key SK is $\alpha \in \mathbb{Z}_p$.

Sign($\mathrm{PP}, \mathrm{SK}, M \in \{0,1\}^\ell$*)* The signing algorithm computes the signature as $\sigma = H(M)^\alpha \in G_{k-1}$. This serves as an aggregate signature for the (single element) multiset $S = (\mathrm{VK}, M)$.

Aggregate($\mathrm{PP}, \tilde{S}, S', \tilde{\sigma}, \sigma'$*).* The aggregation algorithm simply computes the output signature σ as $\sigma = \tilde{\sigma} \cdot \sigma'$. The serves as a signature on the multiset $S = \tilde{S} \cup S'$, where $\cup$ is a *multiset union*.

Verify(PP, S, σ*).* The verification algorithm parses S as $\{(\mathrm{VK}_1, M_1), \ldots, (\mathrm{VK}_{|S|}, , M_{|S|})\}$. It then checks that $e(\sigma, g) \overset{?}{=} \prod_{i=1,\ldots,|S|} e(H(M_i), \mathrm{VK}_i)$ and accepts if and only if it holds.

Correctness To see correctness, an aggregate σ on $S = \{(\mathrm{VK}_1, M_1), \ldots, (\mathrm{VK}_{|S|}, M_{|S|}\}$ is the product of individual signatures; i.e., $\sigma = \prod_{i=1}^{|S|} H(M_i)^{\alpha_i}$ where $\mathrm{VK}_i = g^{\alpha_i}$, and thus passes the verification equation as:

$$\begin{aligned} e(\sigma, g) = e(\prod_{i=1}^{|S|} H(M_i)^{\alpha_i}, g) = \prod_{i=1}^{|S|} e(H(M_i)^{\alpha_i}, g) = \prod_{i=1}^{|S|} e(H(M_i), g)^{\alpha_i} \\ = \prod_{i=1}^{|S|} e(H(M_i), g^{\alpha_i}) = \prod_{i=1}^{|S|} e(H(M_i), VK_i). \end{aligned}$$

Efficiency and Tradeoffs An aggregate signature is one group element in $\mathbb{G}_{k-1}$ independent of the number of messages aggregated. In a multilinear setting, the space to represent a group element might grow with k (which is $\ell + 1$). Indeed, this happens in the GGH [19] graded algebra translation. One way to mitigate this is to differ the message alphabet size in a tradeoff of computation versus storage. The above construction uses a binary message alphabet. If it used an alphabet of 2^d symbols, then the aggregate signature could resident in the group $G_{\ell/d}$ with $\ell/d-1$ pairings required to compute it, at the cost of the public parameters requiring $2^d\ell$ group elements in $\mathbb{G}$.

Construction in the GGH Framework We give a translation of the above construction to the GGH [19] framework in the full version of this work [25].

4.2 Security Analysis

Assumption 1 (Multilinear Computational Diffie-Hellman: k-MCDH) *The k-Multilinear Computational Diffie-Hellman (k-MCDH) problem states the following: A challenger runs $\mathcal{G}(1^\lambda, k)$ to generate groups and generators of order p. Then it picks random $c_1, \ldots, c_k \in \mathbb{Z}_p$. The assumption then states that given $g = g_1, g^{c_1}, \ldots, g^{c_k}$ it is hard for any poly-time algorithm to compute $g_{k-1}^{\prod_{j\in[1,k]} c_j}$ with better than negligible advantage (in security parameter λ).*

We say that the k-MCDH assumption holds against subexponential advantage if there exists a universal constant $\epsilon_0 > 0$ such that no polynomial-time algorithm can succeed in the experiment above with probability greater than $2^{-\lambda^{\epsilon_0}}$. In Section 5.3, we will give a variant of the k-MCDH assumption in the approximate multilinear maps setting of GGH [19] that we will call the GGH k-MCDH assumption. We note that the best cryptanalysis available of the GGH framework [19] suggests that the GGH k-MCDH assumption holds against subexponential advantage.

In the full version [25], we show that the basic aggregate signature scheme for message length ℓ in the distinct message unforgeability game is:

– Selectively secure under the (ℓ+1)-Multilinear Computational Diffie-Hellman (MCDH) assumption.
– Fully secure under the $(\ell + 1)$-MCDH assumption against subexponential advantage.
– Fully secure under a non-interactive, parameterized assumption which depends on message length ℓ, the number of adversarial signing queries and the number of messages in the adversary's forgery.

By applying a simple transformation given in the full version [25] which follows from [10, 2], the distinct message requirement can be removed. Without this transformation, there is a simple attack where the attacker sets some $\mathrm{VK}' = \mathrm{VK}^{-1}$ and submits the identity element in $\mathbb{G}_{k-1}$ as an aggregate forgery for $S = \{(\mathrm{VK}, M), (\mathrm{VK}', M)\}$ for any message M of its choosing.

5 Our ID-Based Aggregate Signature Construction

5.1 Generic Multilinear Construction

Authority-Setup($1^\lambda, \ell, n$*)* The trusted setup algorithm is run by the master authority of the ID-based system. It takes as input the security parameter as well the bit-length ℓ of messages and bit-length n of identities. It first runs $\mathcal{G}(1^\lambda, k = \ell + n)$ and outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$, where we let $g = g_1$.

Next, it chooses random elements $(A_{1,0} = g^{a_{1,0}}, A_{1,1} = g^{a_{1,1}}), \ldots, (A_{\ell,0} = g^{a_{\ell,0}}, A_{\ell,1} = g^{a_{\ell,1}}) \in \mathbb{G}_1^2$ and random exponents $(b_{1,0}, b_{1,1}), \ldots, (b_{n,0}, b_{n,1}) \in {\mathbb{Z}_p}^2$. It sets $B_{i,\beta} = g^{b_{i,\beta}}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$. These will be used to define a function $H(\mathcal{I}, M) : \{0,1\}^n \times \{0,1\}^\ell \to \mathbb{G}_k$. Let $m_1, \ldots, m_\ell$ be the bits of message M and $\mathrm{id}_1, \ldots, \mathrm{id}_n$ as the bits of $\mathcal{I}$. It is computed iteratively as

$$H_1(\mathcal{I}, M) = B_{1,\mathrm{id}_1} \quad \text{for } i \in [2, n] \;\; H_i(\mathcal{I}, M) = e(H_{i-1}(\mathcal{I}, M), B_{i,\mathrm{id}_i})$$

$$\text{for } i \in [n+1, n+\ell = k] \;\; H_i(\mathcal{I}, M) = e(H_{i-1}(\mathcal{I}, M), A_{i-n, m_{i-n}}).$$

We define $H(\mathcal{I}, M) = H_{k=\ell+n}(\mathcal{I}, M)$.

The public parameters, PP, consist of the group sequence description plus:

$$(A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell,1}), (B_{1,0}, B_{1,1}), \ldots, (B_{n,0}, B_{n,1})$$

The master secret key MSK includes PP together with the values $(b_{1,0}, b_{1,1}), \ldots, (b_{n,0}, b_{n,1})$.

KeyGen($\mathrm{MSK}, \mathcal{I} \in \{0,1\}^n$*)* The signing key for identity $\mathcal{I}$ is $\mathrm{SK}_\mathcal{I} = g_{n-1}^{\prod_{i\in[1,n]} b_{i,\mathrm{id}_i}} \in G_{n-1}$.

Sign($\mathrm{PP}, \mathrm{SK}_\mathcal{I}, \mathcal{I} \in \{0,1\}^n, M \in \{0,1\}^\ell$*)* The signing algorithm lets temporary variable $D_0 = \mathrm{SK}_\mathcal{I}$. Then for $i = 1$ to ℓ it computes $D_i = e(D_{i-1}, A_{i,m_i}) \in G_{n-1+i}$. The output signature is

$$\sigma = D_\ell = (g_{k-1})^{(\prod_{i\in[1,n]} b_{i,\mathrm{id}_i})(\prod_{i\in[1,\ell]} a_{i,m_i})}.$$

This serves as an ID-based aggregate signature for the (single element) multiset $S = (\mathcal{I}, M)$.

Aggregate($\mathrm{PP}, \tilde{S}, S', \tilde{\sigma}, \sigma'$*).* The aggregation algorithm simply computes the output signature σ as $\sigma = \tilde{\sigma} \cdot \sigma'$. The serves as a signature on the multiset $S = \tilde{S} \cup S'$, where $\cup$ is a multiset union.

Verify(PP, S, σ*).* It parses S as $\{(\mathcal{I}_1, M_1), \ldots, (\mathcal{I}_{|S|}, M_{|S|})\}$. It then accepts if and only if

$$e(\sigma, g) \stackrel{?}{=} \prod_{i=1,\ldots,|S|} H(\mathcal{I}_i, M_i).$$

Correctness and Security. For correctness, an aggregate σ on $S = \{(\mathcal{I}_1, M_1), \ldots, (\mathcal{I}_{|S|}, M_{|S|})$ is the product of individual signatures; i.e., σ_i where $e(\sigma_i, g) = H(\mathcal{I}_i, M_i)$, and thus $\prod_{i=1}^{|S|} e(\sigma_i, g) = e(\prod_{i=1}^{|S|} \sigma_i, g) = e(\sigma, g) = \prod_{i=1}^{|S|} H(\mathcal{I}_i, M_i)$. Proof of the following theorem appears in the full version [25] and is similar to the proof for the GGH translation which we provide shortly in Section 5.3.

Theorem 2 (Selective Security of ID-Based Construction). *The ID-based aggregate signature scheme for message length ℓ and identity length n in Section 5.1 is selectively secure in the unforgeability game in Section 3 under the $(\ell + n)$-MCDH assumption.*

5.2 ID-Based Construction in the GGH Framework

We show how to modify our ID-based construction to use the GGH [19] graded algebras analogue of multilinear maps. Please note that we use the same notation developed in [19], with some minor changes: Firstly, we use the canonical encoding function cenc provided by the GGH framework more than once at each level of the encoding, but only a globally fixed constant number of times per level. This is compatible with the GGH encoding [19], and allows for a simpler exposition of our scheme and proof. Also, **for ease of notation on the reader, we suppress repeated params arguments that are provided to every algorithm.** Thus, for instance, we will write $\alpha \leftarrow \mathsf{samp}()$ instead of $\alpha \leftarrow \mathsf{samp}(\mathsf{params})$. Note that in our scheme, there will only ever be a single uniquely chosen value for params throughout the scheme, so there is no cause for confusion. Finally, we use the variant of the GGH framework with "strong" zero-testing, where the zero test statistically guarantees that a vector is a valid encoding of zero if it passes the zero test. For further details on the GGH framework, please refer to [19]. See also [20] in these proceedings.

Authority-Setup$(1^\lambda, \ell, n)$ The trusted setup algorithm is run by the master authority of the ID-based system. It takes as input the security parameter as well the bit-length ℓ of messages and bit-length n of identities. It then runs $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^{k=\ell+n})$. Recall that params will be implicitly given as input to all GGH-related algorithms below.

Next, it chooses random encodings $a_{i,\beta} = \mathsf{samp}()$ for $i \in [1, \ell]$ and $\beta \in \{0, 1\}$; and random encodings $b_{i,\beta} = \mathsf{samp}()$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$. Then it assigns $A_{i,\beta} = \mathsf{cenc}_1(1, a_{i,\beta})$ for $i \in [1, \ell]$ and $\beta \in \{0, 1\}$; and it assigns $B_{i,\beta} = \mathsf{cenc}_1(1, b_{i,\beta})$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$.

These will be used to compute a function H mapping $\ell + n$ bit strings to level $k - 1$ encodings. Let $m_1, \ldots, m_\ell$ be the bits of M and $\mathrm{id}_1, \ldots, \mathrm{id}_n$ be the bits of $\mathcal{I}$. It is computed iteratively as

$$H_1(\mathcal{I}, M) = B_{1,\mathrm{id}_1} \quad \text{for } i \in [2, n] \quad H_i(\mathcal{I}, M) = H_{i-1}(\mathcal{I}, M) \cdot B_{i,\mathrm{id}_i}$$

$$\text{for } i \in [n+1, n+\ell = k] \quad H_i(\mathcal{I}, M) = H_{i-1}(\mathcal{I}, M) \cdot A_{i-n, m_{i-n}}.$$

We define $H(\mathcal{I}, M) = \mathsf{cenc}_2(k, H_{k=\ell+n}(\mathcal{I}, M))$.

The public parameters, PP, consist of the $\mathsf{params}, \mathbf{p}_{zt}$ plus:

$$(A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell,1}), (B_{1,0}, B_{1,1}), \ldots, (B_{n,0}, B_{n,1})$$

Note that params includes a level 1 encoding of 1, which we denote as g.

The master secret key MSK includes PP together with the encodings $(b_{1,0}, b_{1,1})$, $\ldots, (b_{n,0}, b_{n,1})$.

KeyGen($\mathrm{MSK}, \mathcal{I} \in \{0,1\}^n$*)* The signing key for identity $\mathcal{I}$ is $\mathrm{SK}_{\mathcal{I}} = \mathsf{cenc}_2(n-1, \prod_{i\in[1,n]} b_{i,\mathrm{id}_i})$.

Sign($\mathrm{PP}, \mathrm{SK}_{\mathcal{I}}, \mathcal{I} \in \{0,1\}^n, M \in \{0,1\}^\ell$*)* The signing algorithm lets temporary variable $D_0 = \mathrm{SK}_{\mathcal{I}}$. Then for $i = 1$ to ℓ it computes $D_i = D_{i-1} \cdot A_{i,m_i}$. The output signature is

$$\sigma = \mathsf{cenc}_3(k-1, D_\ell).$$

This serves as an ID-based aggregate signature for the (single element) multiset $S = (\mathcal{I}, M)$.

Aggregate($\mathrm{PP}, \tilde{S}, S', \tilde{\sigma}, \sigma'$*).* The aggregation algorithm simply computes the output signature σ as $\sigma = \tilde{\sigma} + \sigma'$. The serves as a signature on the multiset $S = \tilde{S} \cup S'$, where $\cup$ is a multiset union.

Verify(PP, S, σ*).* The verification algorithm parses S as $\{(\mathcal{I}_1, M_1), \ldots, (\mathcal{I}_{|S|}, M_{|S|})\}$. It rejects if the multiplicity of any identity/message pair is greater than 2^λ.

The algorithm then proceeds to check the signature by setting $\tau = \mathsf{cenc}_2(1, g)$, and testing: :

$$\mathsf{isZero}\left(\mathbf{p}_{zt}, \tau \cdot \sigma - \sum_{i=1,\ldots,|S|} H(\mathcal{I}_i, M_i)\right)$$

and accepts if and only if the zero testing procedure outputs true. Recall that g above is a canonical level 1 encoding of 1 that is included in params, part of the public parameters.

Correctness. Correctness follows from the same argument as for the ID-based aggregate signature scheme in the generic multilinear setting.

5.3 Proof of Security for ID-based Aggregate Signatures in the GGH framework

We now describe how to modify our proof of security for our ID-based construction to use the GGH [19] graded algebras analogue of multilinear maps. As before, for ease of notation on the reader, we suppress repeated params arguments that are provided to every algorithm. For further details, please see [19].

We begin by describing the GGH analogue of the k-MCDH assumption that we will employ:

Assumption 3 (GGH analogue of k-MCDH: GGH k-MCDH) *The GGH k-Multilinear Computational Diffie-Hellman (GGH k-MCDH) problem states the following: A challenger runs* $\mathsf{InstGen}(1^\lambda, 1^k)$ *to obtain* $(\mathsf{params}, \mathbf{p}_{zt})$*. Note that* params *includes a level 1 encoding of 1, which we denote as g. Then it picks random $c_1, \dots, c_k$ each equal to the result of a fresh call to* $\mathsf{samp}()$*.*

The assumption then states that given $\mathsf{params}, \mathbf{p}_{zt}, \mathsf{cenc}_1(1, c_1), \dots, \mathsf{cenc}_1(1, c_k)$ *it is hard for any poly-time algorithm to compute an integer $t \in [1, 2^\lambda]$ and an encoding z such that*

$$zTst\left(\mathbf{p}_{zt}, \mathsf{cenc}_2(1, g) \cdot z - \mathsf{cenc}_1(k, t \cdot \prod_{j \in [1,k]} c_j)\right)$$

outputs true.

We say the GGH k-MCDH assumption holds against subexponential advantage if there exists a universal constant $\epsilon_0 > 0$ such that no polynomial-time algorithm can succeed in the experiment above with probability greater than $2^{\lambda^{\epsilon_0}}$. The best cryptanalysis available of the GGH framework [19] suggests that the GGH k-MCDH assumption holds against subexponential advantage.

We establish full security of our ID-based aggregate signature scheme conditioned on the k-MCDH assumption holding against subexponential advantage. This follows immediately from the following theorem and a standard complexity leveraging argument:

Theorem 4 (Selective Security of GGH ID-Based Construction). *The ID-based aggregate signature scheme for message length ℓ and identity length n in Section 5.2 is selectively secure in the unforgeability game in Section 3 under the GGH $(\ell + n)$-MCDH assumption.*

Corollary 1. *The ID-based aggregate signature scheme for message length ℓ in Section 5.2 is fully secure in the distinct message unforgeability game under the GGH $(\ell + n)$-MCDH assumption against subexponential advantage.*

Proof. This follows immediately from a complexity leveraging argument: the security parameter λ is chosen to ensure that $2^{\lambda^{\epsilon_0}} >> 2^\ell$, where $2^{-\lambda^{\epsilon_0}}$ is the maximum probability of success allowed in the k-MCDH assumption against subexponential advantage. Now, to establish full security, the simulator performs exactly as in the selective security proof, but first it simply guesses the message that will be forged (instead of expecting the adversary to produce this message). Because this guess will be correct with probability at least $2^{-\ell}$, and the security parameter λ is chosen carefully, full security with polynomial advantage (or even appropriately defined subexponential advantage) implies an attacker on the GGH k-MCDH assumption with subexponential advantage.

Proof. **(of Theorem 4)** We show that if there exists a PPT adversary $\mathcal{A}$ that can break the selective security of the ID-based aggregate signature scheme in the unforgettability game with probability ϵ for message length ℓ, identity length n and security parameter λ, then there exists a PPT simulator that can break the GGH $(\ell + n)$-MCDH assumption for security parameter λ with probability ϵ. The simulator takes as input a GGH MCDH instance $\mathsf{params}, \mathbf{p}_{zt}, C_1 = \mathsf{cenc}_1(1, c_1), \ldots, C_k = \mathsf{cenc}_1(1, c_k)$ where $k = \ell + n$. Let m_i denote the ith bit of M and id_i denote the ith bit of $\mathcal{I}$. The simulator plays the role of the challenger in the game as follows.

Init. Let $\mathcal{I}^* \in \{0,1\}^n$ and $M^* \in \{0,1\}^\ell$ be the forgery identity/message pair output by $\mathcal{A}$.

Setup. The simulator chooses random $x_1, \ldots, x_\ell, y_1, \ldots, y_n$ with fresh calls to $\mathsf{samp}()$. For $i = 1$ to ℓ, let $A_{i,m_i^*} = C_{i+n}$ and $A_{i,\bar{m_i^*}} = \mathsf{cenc}_1(1, x_i)$. For $i = 1$ to n, let $B_{i,\mathrm{id}_i^*} = C_i$ and $B_{i,\bar{\mathrm{id}}_i^*} = \mathsf{cenc}_1(1, y_i)$. The parameters are distributed independently and uniformly at random as in the real scheme.

Queries. Conceptually, the simulator will be able to create keys or signatures for the adversary, because his requests will differ from the challenge identity or message in at least one bit. More specifically,

1. Create New Key: The simulator begins with an index $i = 1$ and an empty sequence of index/identity/private key triples T. On input an identity $\mathcal{I} \in \{0,1\}^n$, if $\mathcal{I} = \mathcal{I}^*$, the simulator records $(i, \mathcal{I}^*, \perp)$ in T. Otherwise, the simulator computes the secret key as follows. Let β be the first index such that $\mathrm{id}_i \neq \mathrm{id}_i^*$. Compute $s = \prod_{i=1,\ldots,n \wedge i \neq \beta} B_{i,\mathrm{id}_i}$. Then compute $\mathrm{SK}_{\mathcal{I}} = \mathsf{cenc}_2(n-1, s \cdot y_\beta)$. Record $(i, \mathcal{I}, \mathrm{SK}_{\mathcal{I}})$ in T. Secret keys are well-formed and, due to the rerandomization in the cenc_2 algorithm, are distributed in a manner statistically exponentially close to the keys generated in the real game.
2. Corrupt User: On input an index $i \in [1, |T|]$, the simulator returns to the adversary the triple $(i, \mathcal{I}_i, \mathrm{SK}_{\mathcal{I}_i}) \in T$. It returns an error if T is empty or i is out of range. Recall that i cannot be associated with $\mathcal{I}^*$ in this game.
3. Sign: On input an index $i \in [1, |T|]$ and a message $M \in \{0,1\}^\ell$, the simulator obtains the triple $(i, \mathcal{I}_i, \mathrm{SK}_{\mathcal{I}_i}) \in T$ or returns an error if it does not exist. If $\mathcal{I}_i \neq \mathcal{I}^*$, then the simulator signs M with $\mathrm{SK}_{\mathcal{I}_i}$ in the usual way.
   If $\mathcal{I}_i = \mathcal{I}^*$, then we know $M \neq M^*$. Let β be the first index such that $m_\beta \neq m_\beta^*$. First compute $\sigma' = \prod_{i=1,\ldots,\ell \wedge i \neq \beta} A_{i,m_i}$. Next, compute $\sigma'' = \sigma' \cdot x_i$. Also compute $\gamma = \prod_{i=1,\ldots,n} B_{i,\mathrm{id}_i}$. Finally, compute $\sigma = \mathsf{cenc}_3(k-1, \gamma \cdot \sigma'')$ Return σ to $\mathcal{A}$. Signatures are well-formed and, due to the rerandomization in the cenc_3 algorithm, distributed in a manner statistically exponentially close to the keys generated in the real game.

Response. Eventually, $\mathcal{A}$ outputs an aggregate signature σ^* on multiset S^* where $(\mathcal{I}^*, M^*) \in S^*$. The simulator will extract from this a solution to the MCDH problem. This works by iteratively computing all the other signatures in S^* and then subtracting them out of the aggregate until only one or more signatures on $(\mathcal{I}^*, M^*)$ remain. That is, the simulator takes an aggregate for S^* and computes an aggregate signature for S' where S' has one less

verification key/message pair than S at each step. These signatures will be computed as in the query phase.
Eventually, we have an aggregate σ' on $t \geq 1$ instances of $(\mathcal{I}^*, M^*)$. However recall that $H(\mathcal{I}^*, M^*)$ is a level k encoding of $(\prod_{i\in[1,n]} b_{i,\mathrm{id}_i^*})(\prod_{i\in[1,\ell]} a_{i,m_i^*}) = \prod_{i\in[k]} c_i$. Thus verification of the signature σ' implies that (t, σ') is a solution to the GGH k-MCDH problem, and so the simulator returns (t, σ') to break the GGH k-MCDH assumption.

The responses of the challenger are distributed statistically exponentially closely to the real unforgeability game. The simulator succeeds whenever $\mathcal{A}$ does.

References

1. Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In *ACM Conference on Computer and Communications Security*, pages 473–484, 2010.
2. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In *ICALP*, pages 411–422, 2007.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
4. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, pages 31–46, 2003.
5. Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *ACM Conference on Computer and Communications Security*, pages 276–285, 2007.
6. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.
7. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.
8. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
9. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. extended abstract in Crypto 2001.
10. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.
11. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.
12. Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.
13. Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In *ASIACRYPT*, pages 644–662, 2012.
14. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

15. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
16. Ying-Ju Chi, Ricardo Oliveira, and Lixia Zhang. Cyclops: The Internet AS-level Observatory. In *ACM SIGCOMM CCR*, 2008.
17. Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. On the instantiability of hash-and-sign rsa signatures. In *TCC*, pages 112–132, 2012.
18. Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In *CRYPTO*, pages 449–466, 2005.
19. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. In *EUROCRYPT*, 2013.
20. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.
21. Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.
22. Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography*, pages 257–273, 2006.
23. Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.
24. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *J. Cryptology*, 25(3):484–527, 2012.
25. Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. Full version available at the Cryptology ePrint Archive `http://eprint.iacr.org/`, 2013.
26. Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.
27. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, pages 465–485, 2006.
28. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT*, pages 74–90, 2004.
29. Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2001.
30. Moni Naor and Omer Reingold. Constructing pseudo-random permutations with a prescribed structure. *J. Cryptology*, 15(2):97–102, 2002.
31. Gregory Neven. Efficient sequential aggregate signed data. *IEEE Transactions on Information Theory*, 57(3):1803–1815, 2011.
32. Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the fiat-shamir scheme. In *ASIACRYPT*, pages 139–148, 1991.
33. Tatsuaki Okamoto. A digital multisignature schema using bijective public-key cryptosystems. *ACM Trans. Comput. Syst.*, 6(4):432–441, 1988.
34. Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In *ISA*, pages 750–759, 2009.
35. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
36. Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.
37. Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.