Unrestricted Identity-Based Aggregate Signcryption in the Standard Model from Multilinear Maps

Hao Wang[1,2,3⋆]

[1] School of Information Science and Engineering, Shandong Normal University
[2] Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology
[3] Shandong Provincial Key Laboratory of Software Engineering

Abstract. Signcryption is a relatively new cryptographic technique that is supposed to fulfill the functionalities of digital signature and encryption in a single logical step and can effectively decrease the computational costs and communication overheads in comparison with the traditional signature-then-encryption schemes, and aggregate signcryption scheme allows individual signcryption ciphertexts intended for the same recipient to be aggregated into a single (shorter) combined ciphertext without losing any of the security guarantees. In this paper, we present a new identity-based aggregate signcryption scheme using multilinear maps. To the best of my knowledge, our new scheme is the first identity-based aggregate signcryption scheme that admits unrestricted aggregation.

1 Introduction

The two most important functionalities offered by cryptography are authentication and confidentiality. In 1997 Zheng introduced the concept of signcryption which provides both confidentiality and authentication [1]. Signcryption is more efficient than performing sign and encrypt independently on a single message using the most efficient signing and encryption algorithms. The first formal security model and security proof was given by Baek et al. in 2002 [2]. Identity based signcryption (IBSC) with formal security proof was introduced by Malone-Lee in his paper [3]. But Malone-Lees scheme was not secure and its weakness was pointed out by Libert and Quisquater in [4]. Libert and Quisquater also proposed three different types of IBSC schemes which satisfy either public verifiability or forward security. It became an open challenge to design an efficient signcryption scheme providing both public verifiability and forward security. Soon, this open problem was solved. Chow et al. [5] designed an IBSC scheme that provides both public verifiability and forward security. Boyen [6] presented an IBSC scheme that provides not only public verifiability and forward security but also ciphertext unlinkability and anonymity. Chen and Malone-Lee [7] improved Boyens scheme in efficiency and Barreto et al. [8] constructed the most efficient IBSC scheme to date. There are two major constraints to design an efficient signcryption scheme, namely computation and communication efficiency. With the increasing running speed of the chips, computational efficiency is not of a serious issue, but the bandwidth is still a limitation. Thus communication efficiency is very important in the present scenario. The

⋆ Corresponding author. Email address: whatsdnu@gmail.com

amount of data sent must be kept as close to the theoretical minimum for getting efficiency. For example in banking scenarios one may have to process many signcryption quickly and simultaneously. In order to reduce the compute cost and overhead of transmission, we use aggregate signcryption. Aggregate signcryption schemes take the conceptual ideas of aggregate signature schemes and apply them to signcryption schemes. An aggregate signcryption scheme allows individual signcryption ciphertexts intended for the same recipient to be aggregated into a single (shorter) combined ciphertext without losing any of the security guarantees that would be present if the original signcryption ciphertexts were transmitted individually. In 2009, Selvi et al. [9] proposed the first identity-based aggregate signcryption (IBASC) along with a formal security model and a formal security proof. Then, there are many researches focus on this topic, such as [10], [11] etc. However, none of those IBASC schemes admits unrestricted aggregation to date.

In this paper, we construct the first identity-based aggregate signcryption scheme that admits unrestricted aggregation, and prove its security in the standard model. We present our results in a generic multilinear map [12] setting and then show how they can be translated to the GGH [13] graded algebras analogue of multilinear maps. This idea is inspired by the related work [14] and [15], the former proposed an identity-based aggregate signatures with unrestricted aggregation and the latter proposed an identity-based key-encapsulation mechanism, both of them are constructed from multilinear maps.

Organization We introduce the leveled multilinear maps, the GGH graded encoding, and the complexity assumption in Section 2, and review the definitions and security model for IBASC in Section 3 & 4. Then, we present our IB-KEM in generic multilinear map setting in Section 5, and make it translated to the GGH framework in Section 6.

2 Leveled Multilinear Maps and the GGH Graded Encoding

2.1 Generic Leveled Multilinear Maps

We give a description of generic, leveled multilinear maps. More details of the GGH graded algebras analogue of mulitlinear maps are included in Appendix A, and for further details, please refer to [13].

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 $\boldsymbol{G} = (\mathbb{G}_1, ..., \mathbb{G}_k)$ 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 $\{e_{i,j} : G_i \times G_j \to G_{i+j} | i, j \geq 1; i + j \leq k\}$. The map $e_{i,j}$ satisfies the following relation:

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

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 $e(g_i^a, g_j^b) = g_{i+j}^{ab}$.

2.2 Algorithmic Components of GGH Encodings

Garg, Gentry and Halevi (GGH) [13] defined an "approximate" version of a multilinear group family, which they call a *graded encoding system*. As a starting point, they view g_i^α in a multilinear group family as simply an *encoding* of α at "level-*i*". This encoding permits basic functionalities, such as equality testing (it is easy to check that two level-*i* encodings encode the same exponent), additive homomorphism (via the group operation in $\mathbb{G}_i$), and bounded multiplicative homomorphism (via the multilinear map *e*). They retain the notion of a somewhat homomorphic encoding with equality testing, but they use probabilistic encodings, and replace the multilinear group family with "less structured" sets of encodings related to lattices.

Abstractly, their *k*-graded encoding system for a ring *R* includes a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,k], \alpha \in R\}$ such that, for every fixed $i \in [0,k]$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint (and thus form a partition of $S_i = \bigcup_\alpha S_i^{(\alpha)}$). The set $S_i^{(\alpha)}$ consists of the "level-*i* encodings of α". Moreover, the system comes equipped with efficient procedures, as follows:

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^k)$ takes as input the security parameter λ and integer k. The procedure outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of an *k*-graded encoding system as above, and $\mathbf{p}_{zt}$ is a level-*k* "zero-test parameter".

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0$, such that the induced distribution on α such that $a \in S_0^{(\alpha)}$ is statistically uniform.

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, i, a)$ takes $i \in [k]$ and a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs a level-*i* encoding $u \in S_i^{(\alpha)}$ for the same α.

Re-Randomization. The randomized $\mathsf{reRand}(\mathsf{params}, i, u)$ re-randomizes encodings to the same level, as long as the initial encoding is under a given noise bound. Specifically, for a level $i \in [k]$ and encoding $u \in S_i^{(\alpha)}$, it outputs another encoding $u' \in S_i^{(\alpha)}$. Moreover for any two encodings $u_1, u_2 \in S_i^{(\alpha)}$ whose noise bound is at most some b, the output distributions of $\mathsf{reRand}(\mathsf{params}, i, u_1)$ and $\mathsf{reRand}(\mathsf{params}, i, u_2)$ are statistically the same.

Addition and negation. Given params and two encodings at the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$, and $\mathsf{neg}(\mathsf{params}, u_1) \in S_i^{(-\alpha_1)}$, subject to bounds on the noise.

Multiplication. For $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$, we have $\mathsf{mult}(\mathsf{params}, u_1, u_2) \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs 1 if $u \in S_k^{(0)}$ and 0 otherwise. Note that in conjunction with the procedure for subtracting encodings, this gives us an

equality test.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-k encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs (say) $K \in \{0,1\}^\lambda$, such that:

- (a) With overwhelming probability over the choice of $\alpha \in R$, for any two $u_1, u_2 \in S_k^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)$,
- (b) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u): \alpha \in R, u \in S_k^{(\alpha)}\}$ is statistically uniform over $\{0,1\}^\lambda$.

The realization method of GGH's graded encoding system is included in Appendix A.

2.3 Complexity Assumption

Assumption 1 (Multilinear Computational Diffie-Hellman: *k*-MCDH) [14] *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 $c_1, ..., c_k \in \mathbb{Z}_p$. The assumption then states that given $g = g_1; g^{c_1}, ..., g^{c_k}$ 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).*

Assumption 2 (GGH analogue of *k*-MCDH: GGH *k*-MCDH) [14] *The GGH k-Multilinear Computational Diffie-Hellman (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 $c_1, ..., c_k$ each equal to the result of a fresh call to $\mathsf{samp}()$.*

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

$$\mathsf{isZero}(\mathbf{p}_{zt}, \mathsf{mult}(g, z) - \mathsf{enc}(k, t \cdot \prod_{j\in[1,k]} c_j))$$

outputs true.

Assumption 3 (Multilinear Decisional Diffie-Hellman: *k*-MDDH) [16] *The k-Multilinear Decisional Diffie-Hellman (k-MDDH) problem states the following: A challenger runs $\mathcal{G}(1^\lambda, k)$ to generate groups and generators of order p. Then it picks random $c_1, ..., c_{k+1} \in \mathbb{Z}_p$. The assumption then states that given $g = g_1, g^{c_1}, ..., g^{c_{k+1}}$ it is hard for any poly-time algorithm to distinguish $g_k^{\prod_{j\in[1,k+1]} c_j}$ from a uniform $\mathbb{G}_k$-element with better than negligible advantage (in security parameter λ).*

Assumption 4 (GGH analogue of *k*-MDDH: GGH *k*-MDDH) [16] *The GGH k-Multilinear Decisional Diffie-Hellman (k-MDDH) 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* $c_1, ..., c_{k+1}$ *each equal to the result of a fresh call to* $\mathsf{samp}()$.

The assumption then states that given params, $\mathbf{p}_{zt}$, $\mathsf{enc}(1, c_1)$, ..., $\mathsf{enc}(1, c_{k+1})$ *and a level-*k *encoding* T*, it is hard for any poly-time algorithm to decide the output of* $\mathsf{isZero}(\mathbf{p}_{zt}, \mathsf{reRand}(T) - \mathsf{reRand}(\mathsf{enc}(\mathsf{params}, k, \prod_{j\in[1,k+1]} c_j)))$ *is 1 or 0 with better than negligible advantage (in security parameter* λ*).*

3 Definitions for Identity-Based Aggregate Signcryption Schemes

We now give our definitions for identity-based aggregate signcryption schemes. In our setting, each signcryption is associated with a *multiset* S over identities. A set S is of the form $\{ID_1, ..., ID_{|S|}\}$. Since S is a multiset it is possible to have $ID_i = ID_j$ for $i \neq j$. All signcryptions, including those that come out of the signcrypt algorithm, are considered to be aggregate signcryptions. The aggregation algorithm is general in that it can take any two aggregate signcryptions and combine them into a new aggregate signcryption.

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. 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 signcryptions into a single one. The aggregation operation does not require any secret keys. The multiset structure allows one to combine two aggregate signcryptions which both signcrypted from the same sender.

An identity-based aggregate signcryption scheme (IBASC) consists of the following five probabilistic polynomial time (PPT) algorithms:

- **Authority-Setup**$(1^\lambda, l, n)$: 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**(*MSK*, $ID \in \{0,1\}^n$) : The key generation algorithm is run by the authority. It takes the master secret key *MSK* and the identity information ID of user $\mathcal{U}_{ID}$ as input and outputs a private key SK_{ID} . The authority sends SK_{ID} to user $\mathcal{U}_{ID}$ through a secure channel.
- **Signcrypt**(*PP*, $M_i \in \{0,1\}^l$, $ID_R \in \{0,1\}^n$, $ID_i \in \{0,1\}^n$, SK_{ID_i}) : For generating the signcryption of a message M_i from user $\mathcal{U}_{ID_i}$ to user $\mathcal{U}_{ID_R}$, the sender $\mathcal{U}_{ID_i}$ provides the system parameters *PP*, the message M_i, the identity information ID_R of receiver $\mathcal{U}_{ID_R}$, the identity information ID_i of its own, and the private key SK_{ID_i} for ID_i as input to this algorithm. The signcryption algorithm outputs the valid signcryption σ_i for the message M_i from user $\mathcal{U}_{ID_i}$ to user $\mathcal{U}_{ID_R}$ and an identities set of senders $S = \{ID_i\}$. *We emphasize that a single signcryption that is output by this algorithm is considered to also be an aggregate signcryption.*
- **Aggregate**(*PP*, S_x, S_y , σ_x, σ_y) : The aggregation algorithm takes as input two multisets S_x and S_y and purported signcryption σ_x and σ_y. The elements of $S_x = \{ID_{x,1}, ...,ID_{x,|S_x|}\}$ consist of the identities corresponding to σ_x and the elements

of $S_y = \{ID_{y,1}, ...,ID_{y,|S_y|}\}$ consist of the identities corresponding to σ_y. The process produces a signcryption σ_{Agg} on the multiset $S_{Agg} = S_x \cup S_y$, where $\cup$ is a multiset union.

– **Unsigncrypt**(*PP*, SK_{ID_R}, S, σ) : The unsigncryption algorithm takes as input the public parameters, the the private key SK_{ID_R} for the receiver with identity ID_R, a multiset $S = \{ID_1, ..., ID_{|S|}\}$ and an aggregate signcryption σ corresponding to S. It outputs a set of messages $\{M_1, ..., M_{|S|}\}$ or a false symbol $\perp$ to indicate whether verification succeeded, where M_i is the message sent from user $\mathcal{U}_{ID_i}$ with identity $ID_i \in S$ to user $\mathcal{U}_{ID_R}$ with identity ID_R.

Correctness The correctness property states that all valid aggregate signcryption will pass the unsigncryption algorithm and output corresponding messages, where a valid aggregate is defined recursively as an aggregate signcryption derived by an application of the aggregation algorithm on two valid inputs or the signcryption algorithm. More formally, for all integers λ, l, n, $k \geq 1$, all *PP*$\in$ **Authority-Setup**($1^\lambda, l, n$), all $ID_1, ..., ID_k \in \{0,1\}^n$, all $SK_{ID_i} \in$ **KeyGen**(*MSK*, ID_i), $\{M_1, ..., M_{|S|}\} \leftarrow$ **Unsigncrypt**(*PP*, SK_{ID_R}, S, σ), if σ is a valid aggregate for multiset S under *PP*. We say that an aggregate signcryption σ is valid for multiset S if: (1) $S = \{ID_i\}$ for some $i \in [1, k]$, and $\sigma \in$ **Signcrypt**(*PP*, $M_i \in \{0,1\}^l$, $ID_R \in \{0,1\}^n$, $ID_i \in \{0,1\}^n$, SK_{ID_i}); or (2) there exists multisets $\widetilde{S}, \widehat{S}$ where $S = \widetilde{S} \cup \widehat{S}$ and valid aggregate signcryption $\widetilde{\sigma}, \widehat{\sigma}$ on them respectively such that $\sigma \in$ **Aggregate**(*PP*, $\widetilde{S}$, $\widehat{S}$, $\widetilde{\sigma}$, $\widehat{\sigma}$).

4 Security Model for Identity Based Aggregate Signcryption Schemes

Security of signcryption consists of two different mechanism: one ensuring authenticity, and the other privacy.

4.1 Unforgeability

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 Π = (**Authority-Setup**, **KeyGen**, **Signcrypt**, **Aggregate**, **Unsigncrypt**).

IBASC-Unforg(Π, $\mathcal{A}$, λ, l, n):

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

– **Queries** : Proceeding adaptively, $\mathcal{A}$ can make four types of requests:

1. **Create New Key** : The challenger begins with an index $i = 1$ and empty sequence of index/identity/private key triples T. On input an identity $ID \in \{0,1\}^n$, the challenger runs **KeyGen**(*MSK*, ID) to obtain SK_{ID}. It adds the

triple (i, ID, SK_{ID}) 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 $(i, ID_i, SK_{ID_i}) \in T$. It returns an error if T is empty or i is out of range.
3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple (i, ID_i, SK_{ID_i})$\in T$ (returning an error if it does not exist) and returns the signcryption resulting from **Signcrypt**(*PP*, M_i, ID_R, ID_i, SK_{ID_i}) to $\mathcal{A}$.
4. **Unsigncrypt** : On input a multiset $S = \{ID_1, ..., ID_{|S|}\}$, an identity ID_R, and an aggregate signcryption σ (from S to ID_R), the challenger obtains the private key SK_{ID_R} for ID_R and returns correspond messages $\{M_1, ..., M_{|S|}\}$ or a false symbol $\perp$ to indicate whether verification succeeded, where M_i is the message sent from user $\mathcal{U}_{ID_i}$ with identity $ID_i \in S$ to user $\mathcal{U}_{ID_R}$ with identity ID_R.

– **Response** : Finally, $\mathcal{A}$ outputs a multiset S^* of identities, a receiver identity ID_R^* and a purported aggregate signcryption σ^*.

We say the adversary "wins" or that the output of this experiment is 1 if: (1) $\{M_1^*$, ..., $M_{|S^*|}^*\} \leftarrow$ **Unsigncrypt**(*PP*, $SK_{ID_R^*}$, S^*, σ^*), (2) there exists an identity $ID_i^* \in S^*$ such that ID_i^* was not queried for a corrupt query by the adversary on any index corresponding to ${ID_i}^*$ and (3) there exists a message M_i^* corresponding to a sender with identity $ID_i^* \in S^*$ such that M_i^* was not queried for a signcrypt query by the adversary on any index corresponding to ${ID_i}^*$. Otherwise, the output is 0. Define **IBASC-Forg**$_{\mathcal{A}}$ as the probability that **IBASC-Unforg**(Π, $\mathcal{A}$, λ, l, n) = 1, where the probability is over the coin tosses of the **Authority-Setup**, **KeyGen**, and **Signcrypt** algorithms and of $\mathcal{A}$.

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

Selective Security We consider a selective variant to **IBASC-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 ($ID^* \in \{0,1\}^n$, $M^* \in \{0,1\}^l$). Note that M^* is the message sent form ID^*. The adversary only "wins" causing the experiment output to be 1 if: (1) $\{M_1^*$, ..., $M_{|S^*|}^*\} \leftarrow$ **Unsigncrypt**(*PP*, $SK_{ID_R^*}$, S^*, σ^*), (2) $M^* \in \{M_1^*$, ..., $M_{|S^*|}^*\}$ is the message corresponding to $ID^* \in S^*$, (3) ID^* was not queried for a corrupt query by the adversary on any index corresponding to ID^* and (4) message M^* was not queried for a signcrypt query by the adversary on any index corresponding to ID^*.

4.2 Confidentiality

Similar to unforgeability game, it should be infeasible for any adversary to distinguish the challenge signcryption ciphertexts, even when the adversary can mount any adaptive chosen identity and chosen ciphertext attacks. This is defined using a game between a challenger and an adversary $\mathcal{A}$ with respect to scheme Π = (**Authority-Setup**, **KeyGen**, **Signcrypt**, **Aggregate**, **Unsigncrypt**).

IBASC-IND-ID-CCA(Π, $\mathcal{A}$, λ, l, n):

- **Setup** : The challenger runs **Authority-Setup**$(1^\lambda, l, n)$ to obtain *PP*. It sends *PP* to $\mathcal{A}$.
- **Phase 1** : Proceeding adaptively, $\mathcal{A}$ can make four types of requests:
  1. **Create New Key** : The challenger begins with an index $i = 1$ and empty sequence of index/identity/private key triples T. On input an identity $ID \in \{0,1\}^n$, the challenger runs **KeyGen**(*MSK*, ID) to obtain SK_{ID}. It adds the triple (i, ID, SK_{ID}) to T and then increments i for the next call. Nothing is returned to the adversary.
  2. **Corrupt User**: On input an index $i \in [1, |T|]$, the challenger returns to the adversary the triple $(i, ID_i, SK_{ID_i}) \in T$. It returns an error if T is empty or i is out of range.
  3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple (i, ID_i, SK_{ID_i})$\in T$ (returning an error if it does not exist) and returns the signcryption resulting from **Signcrypt**(*PP*, M_i, ID_R, ID_i, SK_{ID_i}) to $\mathcal{A}$.
  4. **Unsigncrypt** : On input a multiset $S = \{ID_1, ..., ID_{|S|}\}$, an identity ID_R, and an aggregate signcryption σ (from S to ID_R), the challenger obtains the private key SK_{ID_R} for ID_R and returns correspond messages $\{M_1, ..., M_{|S|}\}$ or a false symbol $\perp$ to indicate whether verification succeeded, where M_i is the message sent from user $\mathcal{U}_{ID_i}$ with identity $ID_i \in S$ to user $\mathcal{U}_{ID_R}$ with identity ID_R.
- **Challenge** : $\mathcal{A}$ chooses a multiset of identities S^* = $\{ID_1^*, ..., ID_{|S^*|}^*\}$, a receiver identity ID_R^*, and two multisets of messages $\mathcal{M}_1^* = \{M_{1,1}^*, ..., M_{1,|S^*|}^*\}$ and $\mathcal{M}_2^* = \{M_{2,1}^*, ..., M_{2,|S^*|}^*\}$, where $ID_R^* \notin S^*$, and $|M_{1,i}^*| = |M_{2,i}^*|$, for $\forall i \in [1, |S^*|]$. The Challenger creates and corrupts all identities in S^*. Then, it chooses a random bit $\beta \in \{0,1\}$, and computes the the signcryptions from **Signcrypt**(*PP*, $M_{\beta,i}^*$, ID_R^*, ID_i^*, $SK_{ID_i^*}$) for $\forall i \in [1, |S^*|]$. Finally, the challenger aggregates these signcryptions using the aggregation algorithm, and returns the result $\{\sigma^*, S^*\}$ to $\mathcal{A}$. The adversarys choice of ID_R^* is restricted to the identities that he did not corrupt in Phase 1.
- **Phase 2** : Phase 1 is repeated with the restriction that the adversary cannot corrupt the identity ID_R^*, and cannot unsigncrypt $\{\sigma^*, S^*\}$.
- **Guess** : $\mathcal{A}$ outputs a bit b'.

$\mathcal{A}$ wins the game if $b' = b$. The advantage of $\mathcal{A}$ is given by **ADV**$_\mathcal{A}$ = $|Pr[b' = b] - \frac{1}{2}|$.

Definition 2. (IND-ID-CCA) *An ID-based aggregate signcryption scheme Π is indistinguishable with respect to adaptive chosen-identity and chosen-ciphertext attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage* $\mathbf{ADV}_{\mathcal{A}}$ *is negligible in λ.*

Chosen-Plaintext Attack We consider a restricted variant to **IBASC-IND-ID-CCA** where the adversary $\mathcal{A}$ is not allowed to make **Unsigncrypt** requests in Phase 1 and Phase 2. We call this **IBASC-IND-ID-CPA** game.

Definition 3. (IND-ID-CPA) *An ID-based aggregate signcryption scheme Π is indistinguishable with respect to adaptive chosen-identity and chosen-plaintext attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage* $\mathbf{ADV}_{\mathcal{A}}$ *is negligible in λ in* **IBASC-IND-ID-CPA** *game.*

Selective Security We consider a selective variant to **IBASC-IND-ID-CPA** (selective in receiver identity) where there is an **Init** phase before the **Setup** phase, wherein $\mathcal{A}$ gives to challenger the receiver identity $ID_R^* \in \{0,1\}^n$. We call this **IBASC-IND-sID-CPA** game.

Definition 4. (IND-sID-CPA) *An ID-based aggregate signcryption scheme Π is indistinguishable with respect to selective chosen-identity and chosen-plaintext attacks if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage* $\mathbf{ADV}_{\mathcal{A}}$ *is negligible in λ in* **IBASC-IND-sID-CPA** *game.*

5 Our Identity-Based Aggregate Signcryption Construction

5.1 Generic Multlinear Construction

Setup(1^λ, *l*, *n*) $\rightarrow$ (*PP*, *MSK*) : The trusted setup algorithm is run by PKG, 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 $\mathcal{G}(1^\lambda, k = l + n)$ and outputs a sequence of groups $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, ..., \mathbb{G}_k)$ of prime order *p*, with canonical generators $g_1, ..., g_k$, where we let $g = g_1$.

Next, it chooses random group elements ($A_{1,0} = g^{a_{1,0}}$, $A_{1,1} = g^{a_{1,1}}$), ..., ($A_{l,0} = g^{a_{l,0}}$, $A_{l,1} = g^{a_{l,1}}$) $\in \mathbb{G}_1^2$. It also chooses random exponents ($b_{1,0}$, $b_{1,1}$), ..., ($b_{n,0}$, $b_{n,1}$) $\in \mathbb{Z}_p^2$ and sets $B_{i,\beta} = g^{b_{i,\beta}}$ for $i \in [1, n]$ and $\beta \in \{0, 1\}$.

These will be used to define two functions $\bar{H}(ID, M) : \{0,1\}^n \times \{0,1\}^l \rightarrow \mathbb{G}_k$ and $\widetilde{H}(ID) : \{0,1\}^n \rightarrow \mathbb{G}_n$. Let $m_1, ..., m_l$ be the bits of message M and $id_1, ..., id_n$ as the bits of ID. It is computed iteratively as

$$\bar{H}_1(ID, M) = \widetilde{H}_1(ID) = B_{1,id_1}$$

$$for\ i \in [2, n]\ \bar{H}_i(ID, M) = e(\bar{H}_{i-1}(ID, M), B_{i,id_i})\ and\ \widetilde{H}_i(ID) = e(\widetilde{H}_{i-1}(ID), B_{i,id_i})$$

$$for\ i \in [n+1, n+l=k]\ \bar{H}_i(ID, M) = e(\bar{H}_{i-1}(ID, M), A_{i-n,m_{i-n}})$$

It defines $\bar{H}(ID, M) = \bar{H}_{k=n+l}(ID, M)$, $\widetilde{H}(ID) = \widetilde{H}_n(ID)$.

Then, it sets a randomness extractor, which can extract a uniform random l bits string form $\mathbb{G}_k$, i.e. $\mathsf{ext} : \mathbb{G}_k \rightarrow \{0,1\}^l$.

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

$$(A_{1,0}, A_{1,1}), ..., (A_{l,0}, A_{l,1}), (B_{1,0}, B_{1,1}), ..., (B_{n,0}, B_{n,1}), \mathsf{ext}$$

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

KeyGen(***MSK***, $ID \in \{0,1\}^n$) : The private key for identity $ID = (id_1, ..., id_n)$ is $SK_{ID} = g_{n-1}^{\prod_{j\in[1,n]} b_{j,id_j}} \in \mathbb{G}_{n-1}$.

Signcrypt($M_i \in \{0,1\}^l$, $ID_R \in \{0,1\}^n$, $ID_i \in \{0,1\}^n$, SK_{ID_i}, ***PP***) : The Signcryption algorithm takes $M_i = (m_{i,1}, ..., m_{i,l})$, $ID_R = (id_{R,1}, ..., id_{R,n})$, $ID_i = (id_{i,1}, ..., id_{i,n})$, $SK_{ID_i} = g_{n-1}^{\prod_{j\in[1,n]} b_{j,id_{i,j}}}$ and ***PP*** as input, lets temporary variable $D_{i,0} = SK_{ID_i}$, and chooses $t_i \in \mathbb{Z}_p$ randomly. Then for $j = 1$ to l it computes $D_{i,j} = e(D_{j-1}, A_{j,m_{i,j}}) \in \mathbb{G}_{n-1+i}$. The output signcryption is $\sigma_i = (\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2})$, where

$$\sigma_{i,0} = \mathsf{ext}(e(\widetilde{H}(ID_R), g_l^{t_i})) \oplus M_i, \;\; \sigma_{i,1} = g_{l+1}^{t_i},$$

$$\sigma_{i,2} = D_l = (g_{k-1})^{\prod_{j\in[1,n]} b_{j,id_{i,j}} \cdot \prod_{j\in[1,l]} a_{i,m_{i,j}}}.$$

This serves as an ID-based aggregate signcryption for the (single element) multiset $S = \{ID_i\}$.

Aggregate(***PP***, S_x, S_y , σ_x, σ_y) : The aggregation algorithm simply computes the output signcryption σ_z as $\sigma_z = (\sigma_{x,0} \| \sigma_{y,0}, \sigma_{x,1} \| \sigma_{y,1}, \sigma_{x,2} \cdot \sigma_{y,2})$, where $\|$ is a concatenation symbol. The serves as a signcryption on the multiset $S_z = S_x \cup S_y$, where $\cup$ is a multiset union.

Unsigncrypt(***PP***, SK_{ID_R}, S, σ) : For $S = \{ID_1, ..., ID_{|S|}\}$, $\sigma = \{ \sigma_{ID_1,0} \| ... \| \sigma_{ID_{|S|},0}, \sigma_{ID_1,1} \| ... \| \sigma_{ID_{|S|},1}, \sigma_2 \}$, the unsigncryption algorithm computes that

$$M_i = \sigma_{ID_i,0} \oplus \mathsf{ext}(e(SK_{ID_R}, \sigma_{ID_i,1})) \quad for \;\; i \in \{1, ..., |S|\}$$

It then accepts if and only if

$$e(\sigma_2, g) \stackrel{?}{=} \prod_{i=1,...,|S|} \bar{H}(ID_i, M_i).$$

5.2 Correctness

For correct "decryption", we have

$$\begin{aligned}\widetilde{H}(ID_R) &= \widetilde{H}_n(ID_R)\\ &= e(\widetilde{H}_{n-1}(ID), B_{n,id_{R,n}})\\ &= e(e(\widetilde{H}_{n-2}(ID), B_{n-1,id_{R,n-1}}), B_{n,id_{R,n}})\\ &.....\\ &= e(B_{1,id_{R,1}}, ..., B_{n,id_{R,n}})\\ &= e(g_1^{b_{1,id_{R,1}}}, ..., g_1^{b_{n,id_{R,n}}})\\ &= g_n^{\prod_{j\in[1,n]} b_{j,id_{R,j}}}\end{aligned}$$

$$\begin{aligned}e(SK_{ID_R}, \sigma_{ID_i,1}) &= e(g_{n-1}^{\prod_{j\in[1,n]} b_{j,id_{R,j}}}, g_{l+1}^{t_i})\\ &= g_{n+l}^{\prod_{j\in[1,n]} b_{j,id_{R,j}}\cdot t_i}\end{aligned}$$

Therefore, for $i \in [1, |S|]$:

$$e(\widetilde{H}(ID_R), g_l^{t_i}) = g_{n+l}^{\prod_{j\in[1,n]} b_{j,id_{R,j}}\cdot t_i} = e(SK_{ID_R}, \sigma_{ID_i,1})$$

Then,

$$\sigma_{ID_i,0} \oplus \mathsf{ext}(e(SK_{ID_R}, \sigma_{ID_i,1})) = M_i$$

For "authentication", we have

$$\begin{aligned}\sigma_2 &= \prod_{i\in[1,|S|]} \sigma_{i,2}\\ &= \prod_{i\in[1,|S|]} (g_{k-1})^{\prod_{j\in[1,n]} b_{j,id_{i,j}}\cdot\prod_{j\in[1,l]} a_{j,m_{i,j}}}\\ &= (g_{k-1})^{\sum_{i\in[1,|S|]}(\prod_{j\in[1,n]} b_{j,id_{i,j}}\cdot\prod_{j\in[1,l]} a_{j,m_{i,j}})}\end{aligned}$$

$$\begin{aligned}\prod_{i=1,...,|S|} \bar{H}(ID_i, M_i) &= \prod_{i=1,...,|S|} (g_k)^{\prod_{j\in[1,n]} b_{j,id_{i,j}}\cdot\prod_{j\in[1,l]} a_{j,m_{i,j}}}\\ &= (g_k)^{\sum_{i\in[1,|S|]}(\prod_{j\in[1,n]} b_{j,id_{i,j}}\cdot\prod_{j\in[1,l]} a_{j,m_{i,j}})}\end{aligned}$$

Therefore, if (S, σ) is a "correct" aggregate signcryption ciphertext, then

$$e(\sigma_2, g) = \prod_{i=1,...,|S|} \bar{H}(ID_i, M_i) = (g_k)^{\sum_{i\in[1,|S|]}(\prod_{j\in[1,n]} b_{j,id_{i,j}}\cdot\prod_{j\in[1,l]} a_{j,m_{i,j}})}.$$

5.3 Security

Unforgeability

Theorem 1. *The ID-based aggregate signcryption scheme for message length l and identity length n in Section 5.1 is selectively secure in the unforgeability game in Section 4 under the $(l+n)$-MCDH assumption.*

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ that can break the selective security of the ID-based aggregate signcryption scheme in the unforgeability game with a non-negligible advantage for message length l, identity length n and security parameter λ, then there exists a PPT simulator that can break the $(l+n)$-MCDH assumption. The simulator takes as input a MCDH instance $(g, g^{c_1}, ..., g^{c_k})$ together with the group descriptions where $k = l + n$. Let m_j denote the *j*-th bit of M and id_j denote the *j*-th bit of ID. The simulator plays the role of the challenger in the game as follows.

- ***Init***: Let $ID^* \in \{0,1\}^n$ and $M^* \in \{0,1\}^l$ be the forgery identity/message pair output by $\mathcal{A}$.
- ***Setup***: The simulator chooses random $x_1, ..., x_l, y_1, ..., y_n \in \mathbb{Z}_p$. For i = 1 to l, let $A_{i,m_i^*} = g^{c_{i+n}}$ and $A_{i,1-m_i^*} = g^{x_i}$. For i = 1 to n, let $B_{i,id_i^*} = g^{c_i}$ and $B_{i,1-id_i^*} = g^{y_i}$. We remark that 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 signcrypt or unsigncrypt 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 $ID \in \{0,1\}^n$, if $ID = ID^*$, the simulator records (i, ID^*, $\perp$) in T. Otherwise, the simulator computes the secret key as follows. Let β be the first index such that $id_\beta \neq id_\beta^*$. Use $n-2$ pairings on the B_{j,id_j} values to compute $s = (g_{n-1})^{\prod_{j=1,...,n \wedge j \neq \beta} b_{j,id_j}}$. Then compute $SK_{ID} = s^{y_\beta} = (g_{n-1})^{\prod_{j=1,...,n} b_{j,id_j}}$. Record ($i$, ID, SK_{ID}) in T. Secret keys are unique and perfectly distributed as in the real game.
  2. **Corrupt User**: On input an index $i \in [1, |T|]$, the simulator returns to the adversary the triple (i, ID_i, SK_{ID_i}) $\in T$. It returns an error if T is empty or i is out of range. Recall that i cannot be associated with ID^* in this game.
  3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple (i, ID_i, SK_{ID_i})$\in T$ or returns an error if it does not exist. If $ID_i \neq ID^*$, then the simulator creates signcryption ciphertext in the usual way.
     If $ID_i = ID^*$, then we know $M_i \neq M^*$. The simulator creates $\sigma_{i,0}$ and $\sigma_{i,1}$ in the usual way. Let β be the first index such that $m_{i,\beta} \neq m_\beta^*$. Use $l-2$ pairings on the $A_{j,m_{i,j}}$ values to compute $\sigma'_{i,2} = (g_{l-1})^{\prod_{j=1,...,l \wedge j \neq \beta} a_{j,m_{i,j}}}$. Next, compute $\sigma''_{i,2} = {\sigma'_{i,2}}^{x_i} = (g_{l-1})^{\prod_{j=1,...,l} a_{j,m_{i,j}}}$. Use $n-1$ pairings on the $B_{j,id_{i,j}}$ values to compute $\gamma = (g_n)^{\prod_{j=1,...,n} b_{j,id_{i,j}}}$. Finally, compute $\sigma_{i,2} = e(\gamma, \sigma''_{i,2}) = (g_{k-1})^{\prod_{j \in [1,n]} b_{j,id_{i,j}} \cdot \prod_{j \in [1,l]} a_{i,m_{i,j}}}$. Return $\sigma_i =$

$(\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2})$ and $S = \{ID_i\}$ to $\mathcal{A}$. Signcryption ciphertexts are unique and perfectly distributed as in the real game.

4. **Unsigncrypt** : The simulator can run unsigncryption in the usual way, because it can create and corrupt any identity $ID_R \neq ID^*$ freely.

– **Response**: $\mathcal{A}$ outputs an aggregate signcryption ciphertext σ^* on multiset S^* where $ID^* \in S$ and M^* is the corresponding message sent from ID^*. 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 dividing them out of the aggregate until only one or more signcryption ciphertexts on (ID^*, M^*) remain. That is, the simulator takes an aggregate for S^* and computes an aggregate signcryption ciphertext for S' where S' has one less identity than S^* at each step. These signatures will be computed as in the query phase.
Eventually, we have an aggregate instances σ' on $t \geq 1$ of (ID^*,M^*). We have that $e(\sigma'_2, g) = \bar{H}(ID^*, M^*)^t = (g_k)^{t \cdot (\prod_{j \in [1,n]} b_{j,id_j^*} \cdot \prod_{j \in [1,l]} a_{j,m_j^*})}$ and thus $\sigma'_2 = (g_{k-1})^{t \prod_{j \in [1,k]} c_j}$. The simulator computes $\sigma_2'^{1/t}$(recall that t is not 0 mod p) which gives $(g_{k-1})^{\prod_{j \in [1,k]} c_j}$ and this is given as the solution to the MCDH problem.

As remarked in the Setup and Query phase, the responses of the challenger are distributed identically to the real unforgeability game. The simulator succeeds whenever $\mathcal{A}$ does.

Confidentiality

Theorem 2. *The ID-based aggregate signcryption scheme for message length l and identity length n in Section 5.1 is IND-sID-CPA under the $(l+n)$-MDDH assumption.*

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ that can break the IND-sID-CPA security of the ID-based aggregate signcryption scheme in **IBASC-IND-sID-CPA** game with a non-negligible advantage for message length l, identity length n and security parameter λ, then there exists a PPT simulator $\mathcal{B}$ that can break the $(l+n)$-MDDH assumption. The simulator takes as input a MDDH instance $(g, g^{c_1}, ..., g^{c_{k+1}}, T)$ together with the group descriptions, where $k = l + n$, and T is identical to $g_k^{\prod_{j \in [1,k+1]} c_j}$ or uniform and independent in $\mathbb{G}_k$. $\mathcal{B}$'s goal is to output 1 if $T = g_k^{\prod_{j \in [1,k+1]} c_j}$ and 0 otherwise. Let m_j denote the *j*-th bit of M and id_j denote the *j*-th bit of ID. The simulator plays the role of the challenger in the game as follows.

– ***Init***: $\mathcal{A}$ outputs an identity $ID_R^* = (id_{R,1}^*, ..., id_{R,n}^*)$, where it wishes to be challenged, the $id_{R,j}^*$ is the *j*-th bit of ID_R^*.
– ***Setup***: $\mathcal{B}$ chooses random group elements ($A_{1,0} = g^{a_{1,0}}$, $A_{1,1} = g^{a_{1,1}}$), ..., ($A_{l,0} = g^{a_{l,0}}$, $A_{l,1} = g^{a_{l,1}}$) $\in \mathbb{G}_1^2$. It also chooses random exponents b_1,..., $b_n \in \mathbb{Z}_p$ and sets ($B_{i,id_i^*} = g^{c_i}$, $B_{i,(1-id_i^*)} = g^{b_i}$) for $i \in [1, n]$. Then, it sets a randomness extractor $\mathsf{ext} : \mathbb{G}_k \rightarrow \{0,1\}^l$, and sends $(A_{1,0}, A_{1,1})$, ..., $(A_{l,0}, A_{l,1})$,$(B_{1,0}, B_{1,1})$, ..., $(B_{n,0}, B_{n,1})$, ext as well as the group sequence description to $\mathcal{A}$. We remark that the parameters are distributed independently and uniformly at random as in the real scheme.

- ***Phase 1 & 2***: Conceptually, the simulator will be able to create keys or signcrypt for the adversary, because his requests will differ from the challenge identity 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 $ID \in \{0,1\}^n$, if $ID = ID^*_R$, the simulator records (i, ID^*_R, $\perp$) in T. Otherwise, the simulator computes the secret key as follows. Let β be the first index such that $id_\beta \neq id^*_{R,\beta}$. Use $n-2$ pairings on the B_{j,id_j} values to compute $s = (g_{n-1})^{\prod_{j=1,\ldots,n \wedge j \neq \beta} b_{j,id_j}}$. Then compute $SK_{ID} = s^{b_\beta} = (g_{n-1})^{\prod_{j=1,\ldots,n} b_{j,id_j}}$. Record ($i$, ID, SK_{ID}) in T. Secret keys are unique and perfectly distributed as in the real game.
  2. **Corrupt User**: On input an index $i \in [1, |T|]$, the simulator returns to the adversary the triple (i, ID_i, SK_{ID_i}) $\in T$. It returns an error if T is empty or i is out of range. Recall that i cannot be associated with ID^*_R in this game.
  3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple (i, ID_i, SK_{ID_i})$\in T$ or returns an error if it does not exist. Then, the simulator creates signcryption ciphertext in the usual way.
- ***Challenge***: $\mathcal{A}$ chooses a multiset of identities S^* = $\{ID^*_1, ..., ID^*_{|S^*|}\}$, and two multisets of messages $\mathcal{M}^*_1 = \{M^*_{1,1}, ..., M^*_{1,|S^*|}\}$ and $\mathcal{M}^*_2 = \{M^*_{2,1}, ..., M^*_{2,|S^*|}\}$. $\mathcal{B}$ creates and corrupts all identities in S^*, and gets $SK_{ID^*_i}$ for $\forall\ ID^*_i \in S^*$. We note, $\mathcal{B}$ can do this, because $ID^*_R \notin S^*$. Then, it chooses a random bit $\beta \in \{0,1\}$, chooses a random integer $t'_i \in \mathbb{Z}_p$, and calculates the the signcryption ciphertexts $\sigma^*_i = (\sigma^*_{i,0}, \sigma^*_{i,1}, \sigma^*_{i,2})$ for $ID^*_i \in S^*$, where $\sigma_{i,2}$ is calculated in the usual way, and

$$\sigma_{i,0} = \mathsf{ext}(T^{t'_i}) \oplus M^*_{\beta,i},\ \ \sigma_{i,1} = g_{l+1}^{t'_i \prod_{j \in [1,l+1]} c_{n+j}}.$$

  Finally, the challenger aggregates these signcryptions using the aggregation algorithm, and returns the result $\{\sigma^*, S^*\}$ to $\mathcal{A}$.
- ***Guess***: $\mathcal{A}$ outputs his guess $b' \in \{0,1\}$ for b.

If $b = 1$ then $\mathcal{A}$ played the proper security game. On the other hand, if $b = 0$, all information about the message M^*_b is lost. Therefore the advantage of $\mathcal{A}$ is exactly 0. As a result if $\mathcal{A}$ breaks the proper security game with a non-negligible advantage, then $\mathcal{B}$ has a non-negligible advantage in breaking the (l+n)-MDDH assumption.

6 IB-ASC in the GGH Framework

In this section, we show how to modify our ID-based construction to use the GGH [13] graded algebras analogue of multilinear maps. 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. For further details on the GGH framework, please refer to [13]. The realization method of GGH's graded encoding system is included in Appendix A.

6.1 Construction in the GGH Framework

Setup(1^λ, l, n): The trusted setup algorithm is run by PKG, 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 (params, $\mathbf{p}_{zt}$)$\leftarrow$$\mathsf{InstGen}(1^\lambda, 1^n)$. Recall that params will be implicitly given as input to all GGH-related algorithms below.

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

These will be used to define two functions $\bar{H}(ID, M) : \{0,1\}^n \times \{0,1\}^l \rightarrow \mathbb{G}_k$ and $\widetilde{H}(ID) : \{0,1\}^n \rightarrow \mathbb{G}_n$. Let m_1, ..., m_l be the bits of message M and id_1, ..., id_n as the bits of ID. It is computed iteratively as

$$\bar{H}_1(ID, M) = \widetilde{H}_1(ID) = B_{1,id_1}$$

$$for\ i \in [2,n]\ \ \bar{H}_i(ID, M) = \mathsf{mult}(\bar{H}_{i-1}(ID, M), B_{i,id_i}),$$

$$\widetilde{H}_i(ID) = \mathsf{mult}(\widetilde{H}_{i-1}(ID), B_{i,id_i})$$

$$for\ i \in [n+1, n+l=k]\ \ \bar{H}_i(ID, M) = \mathsf{mult}(\bar{H}_{i-1}(ID, M), A_{i-n,m_{i-n}})$$

It defines $\bar{H}(ID, M) = \mathsf{reRand}(k, \bar{H}_k(ID, M))$, $\widetilde{H}(ID) = \mathsf{reRand}(n, \widetilde{H}_n(ID))$.

Then, it sets a pseudorandom number generator $\mathcal{F}(s) \rightarrow s'$, where $s \in \{0,1\}^\lambda$ is the seed, and $s' \in \{0,1\}^l$ is a pseudorandom number.

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

$$(A_{1,0}, A_{1,1}), ..., (A_{l,0}, A_{l,1}), (B_{1,0}, B_{1,1}), ..., (B_{n,0}, B_{n,1}), \mathcal{F}$$

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

KeyGen(*MSK*, *ID* $\in \{0,1\}^n$) : The private key for identity $ID = (id_1, ..., id_n)$ is $SK_{ID} = \mathsf{reRand}(n-1, \mathsf{enc}(n-1, \prod_{i\in[1,n]} b_{i,id_i}))$.

Signcrypt($M_i \in \{0,1\}^l$, $ID_R \in \{0,1\}^n$, $ID_i \in \{0,1\}^n$, SK_{ID_i}, *PP*) : The Signcryption algorithm takes $M_i = (m_{i,1}, ..., m_{i,l})$, $ID_R = (id_{R,1}, ..., id_{R,n})$, $ID_i = (id_{i,1}, ..., id_{i,n})$, $SK_{ID_i} = \mathsf{reRand}(n-1, \mathsf{enc}(n-1, \prod_{j\in[1,n]} b_{j,id_{i,j}}))$ and *PP* as input, lets temporary variable $D_{i,0} = SK_{ID_i}$, and chooses encodings $t_i = \mathsf{samp}()$ randomly. Then for $j = 1$ to l it computes $D_{i,j} = \mathsf{mult}(D_{j-1}, A_{j,m_{i,j}}) \in \mathbb{G}_{n-1+i}$. The output signcryption is $\sigma_i = (\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2})$, where

$$\sigma_{i,0} = \mathcal{F}(\mathsf{ext}(\mathsf{mult}(\widetilde{H}(ID_R), \mathsf{enc}(l, t_i)))) \oplus M_i,\ \ \sigma_{i,1} = \mathsf{enc}(l+1, t_i),$$

$$\sigma_{i,2} = \mathsf{reRand}(k-1, D_l).$$

This serves as an ID-based aggregate signcryption for the (single element) multiset $S = \{ID_i\}$.

Aggregate(*PP*, S_x, S_y , σ_x, σ_y) : The aggregation algorithm simply computes the output signcryption σ_z as $\sigma_z = (\sigma_{x,0}\|\sigma_{y,0}, \sigma_{x,1}\|\sigma_{y,1}, \sigma_{x,2} + \sigma_{y,2})$, where $\|$ is a concatenation symbol. The serves as a signcryption on the multiset $S_z = S_x \cup S_y$, where $\cup$ is a multiset union.
Unsigncrypt(*PP*, SK_{ID_R}, S, σ) : For $S = \{ID_1, ..., ID_{|S|}\}$, $\sigma = \{\ \sigma_{ID_1,0}\ \|...\|\ \sigma_{ID_{|S|},0}, \sigma_{ID_1,1}\ \|...\|\ \sigma_{ID_{|S|},1}, \sigma_2\ \}$, the unsigncryption algorithm computes that

$$M_i = \sigma_{ID_i,0} \oplus \mathcal{F}(\mathsf{ext}(\mathsf{mult}(SK_{ID_R}, \sigma_{ID_i,1}))) \quad for \quad i \in \{1, ..., |S|\}$$

It then accepts if and only if the under zero testing procedure outputs true.

$$\mathsf{isZero}(\mathbf{p}_{zt}, \mathsf{mult}(\sigma_2, g) - \sum_{i=1,...,|S|} \bar{H}(ID_i, M_i))$$

Correctness. Correctness follows from the same argument as for the IB-KEM in the generic multilinear setting.

6.2 Security

Unforgeability

Theorem 3. *The ID-based aggregate signcryption scheme for message length l and identity length n in Section 6.1 is selectively secure in the unforgeability game in Section 4 under the GGH $(l + n)$-MCDH assumption.*

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ that can break the selective security of the ID-based aggregate signcryption scheme in the unforgeability game with a non-negligible advantage for message length l, identity length n and security parameter λ, then there exists a PPT simulator that can break the GGH $(l+n)$-MCDH assumption. The simulator takes as input a GGH MCDH instance, params, $\boldsymbol{p}_{zt}$, $C_1 = \mathsf{enc}(1, c_1)$, ..., $C_k = \mathsf{enc}(1, c_k)$ where $k = l + n$. Let m_j denote the *j*-th bit of M and id_j denote the *j*-th bit of ID. The simulator plays the role of the challenger in the game as follows.

- ***Init***: Let $ID^* \in \{0,1\}^n$ and $M^* \in \{0,1\}^l$ be the forgery identity/message pair output by $\mathcal{A}$.
- ***Setup***: The simulator chooses random $x_1, ..., x_l, y_1, ..., y_n$ with fresh calls to $\mathsf{samp}()$. For i = 1 to l, let $A_{i,m_i^*} = C_{i+n}$ and $A_{i,1-m_i^*} = \mathsf{enc}(1, x_i)$. For i = 1 to n, let $B_{i,id_i^*} = C_i$ and $B_{i,1-id_i^*} = \mathsf{enc}(1, y_i)$. We remark that 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 signcrypt or unsigncrypt 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 $ID \in \{0,1\}^n$, if $ID = ID^*$, the simulator records (i, ID^*, $\perp$) in T. Otherwise, the simulator computes the secret key as follows. Let β be the first index such that $id_\beta \neq id_\beta^*$. Use $n - 2$ pairings on the B_{j,id_j} values to compute $s = \prod_{j=1,...,n \wedge j \neq \beta} B_{j,id_j}$. Then compute $SK_{ID} = \mathsf{reRand}(n - 1, s \cdot y_\beta)$. Record ($i$, ID, SK_{ID}) in T. Secret keys are unique and perfectly distributed as in the real game.

2. **Corrupt User**: On input an index $i \in [1, |T|]$, the simulator returns to the adversary the triple $(i, ID_i, SK_{ID_i}) \in T$. It returns an error if T is empty or i is out of range. Recall that i cannot be associated with ID^* in this game.
3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple $(i, ID_i, SK_{ID_i}) \in T$ or returns an error if it does not exist. If $ID_i \neq ID^*$, then the simulator creates signcryption ciphertext in the usual way.
   If $ID_i = ID^*$, then we know $M_i \neq M^*$. The simulator creates $\sigma_{i,0}$ and $\sigma_{i,1}$ in the usual way. Let β be the first index such that $m_{i,\beta} \neq m^*_\beta$. Use $l-2$ pairings on the $A_{j,m_{i,j}}$ values to compute $\sigma'_{i,2} = \prod_{j=1,\ldots,l \wedge j \neq \beta} A_{j,m_{i,j}}$. Next, compute $\sigma''_{i,2} = \sigma'_{i,2} \cdot x_i$. Use $n-1$ pairings on the $B_{j,id_{i,j}}$ values to compute $\gamma = \prod_{j=1,\ldots,n} B_{j,id_{i,j}}$. Finally, compute $\sigma_{i,2} = \mathsf{reRand}(k-1, \gamma \cdot \sigma''_{i,2})$. Return $\sigma_i = (\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2})$ and $S = \{ID_i\}$ to $\mathcal{A}$. Signcryption ciphertexts are unique and perfectly distributed as in the real game.
4. **Unsigncrypt** : The simulator can run unsigncryption in the usual way, because it can create and corrupt any identity $ID_R \neq ID^*$ freely.

– **Response**: $\mathcal{A}$ outputs an aggregate signcryption ciphertext σ^* on multiset S^* where $ID^* \in S$ and M^* is the corresponding message sent from ID^*. The simulator will extract from this a solution to the GGH MCDH problem. This works by iteratively computing all the other signatures in S^* and then dividing them out of the aggregate until only one or more signcryption ciphertexts on (ID^*, M^*) remain. That is, the simulator takes an aggregate for S^* and computes an aggregate signcryption ciphertext for S' where S' has one less identity than S^* at each step. These signatures will be computed as in the query phase.
  Eventually, we have an aggregate instances σ' on $t \geq 1$ of (ID^*,M^*). However recall that $\bar{H}(ID^*, M^*)$ is a level k encoding of $(\prod_{j\in[1,n]} b_{j,id^*_j}) \cdot (\prod_{j\in[1,l]} a_{j,m^*_j})$ $= \prod_{j\in[1,k]} c_j$. Thus verification of the Signcryption ciphertexts implies that (t, σ'_2) is a solution to the GGH k-MCDH problem, and so the simulator returns (t, σ'_2) to break the GGH k-MCDH assumption.

As remarked in the Setup and Query phase, the responses of the challenger are distributed identically to the real unforgeability game. The simulator succeeds whenever $\mathcal{A}$ does.

Confidentiality

Theorem 4. *The ID-based aggregate signcryption scheme for message length l and identity length n in Section 6.1 is IND-sID-CPA under the GGH $(l+n)$-MDDH assumption.*

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ that can break the IND-sID-CPA security of the ID-based aggregate signcryption scheme in **IBASC-IND-sID-CPA** game with a non-negligible advantage for message length l, identity length n and security parameter λ, then there exists a PPT simulator $\mathcal{B}$ that can break the $(l+n)$-MDDH assumption. The simulator takes as input a MDDH instance, params, $\mathbf{p}_{zt}$, $C_1 = \mathsf{enc}(1, c_1)$, ..., $C_{k+1} = \mathsf{enc}(1, c_{k+1})$ and a level-k encoding T, where $k = l + n$.

Algorithm $\mathcal{B}$'s goal is to output 1 if isZero($\boldsymbol{p}_{zt}$, reRand(T)-reRand(enc(params, k, $\prod_{j\in[1,k+1]} c_j$)))=1 and 0 otherwise. Let m_j denote the *j*-th bit of M and id_j denote the *j*-th bit of ID. The simulator plays the role of the challenger in the game as follows.

- ***Init***: $\mathcal{A}$ outputs an identity $ID^*_R = (id^*_{R,1}, ..., id^*_{R,n})$, where it wishes to be challenged, the $id^*_{R,j}$ is the *j*-th bit of ID^*_R.
- ***Setup***: $\mathcal{B}$ chooses random $a_{1,0}, a_{1,1}..., a_{l,0}, a_{l,1}, b_1, ..., b_n$ with fresh calls to samp(), and sets ($A_{i,0}$ = enc$(1, a_{i,0})$, $A_{i,1}$ = enc$(1, a_{i,1})$) for $i \in [1, l]$, sets (B_{i,id^*_i} = C_i, $B_{i,(1-id^*_i)}$ = enc$(1, b_i)$) for $i \in [1, n]$. We remark that the parameters are distributed independently and uniformly at random as in the real scheme.
- ***Phase 1 & 2***: Conceptually, the simulator will be able to create keys or signcrypt for the adversary, because his requests will differ from the challenge identity 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 $ID \in \{0,1\}^n$, if $ID = ID^*_R$, the simulator records (i, ID^*_R, $\perp$) in T. Otherwise, the simulator computes the secret key as follows. Let β be the first index such that $id_\beta \neq id^*_{R,\beta}$. Use $n-2$ pairings on the B_{j,id_j} values to compute $s = \prod_{j=1,...,n \wedge j\neq\beta} B_{j,id_j}$. Then compute $SK_{ID} =$ reRand$(n-1, s \cdot b_\beta)$. Record (i, ID, SK_{ID}) in T. Secret keys are unique and perfectly distributed as in the real game.
  2. **Corrupt User**: On input an index $i \in [1, |T|]$, the simulator returns to the adversary the triple (i, ID_i, SK_{ID_i}) $\in T$. It returns an error if T is empty or i is out of range. Recall that i cannot be associated with ID^*_R in this game.
  3. **Signcrypt**: On input an index $i \in [1, |T|]$, an identity information ID_R of receiver, and a message $M_i \in \{0,1\}^l$, the challenger obtains the triple (i, ID_i, SK_{ID_i})$\in T$ or returns an error if it does not exist. Then, the simulator creates signcryption ciphertext in the usual way.
- ***Challenge***: $\mathcal{A}$ chooses a multiset of identities S^* = $\{ID^*_1, ..., ID^*_{|S^*|}\}$, and two multisets of messages $\mathcal{M}^*_1 = \{M^*_{1,1}, ..., M^*_{1,|S^*|}\}$ and $\mathcal{M}^*_2 = \{M^*_{2,1}, ..., M^*_{2,|S^*|}\}$. $\mathcal{B}$ creates and corrupts all identities in S^*, and gets $SK_{ID^*_i}$ for $\forall\ ID^*_i \in S^*$. We note, $\mathcal{B}$ can do this, because $ID^*_R \notin S^*$. Then, it chooses a random bit $\beta \in \{0,1\}$, chooses random t'_i with fresh calls to samp(), and calculates the the signcryption ciphertexts $\sigma^*_i = (\sigma^*_{i,0}, \sigma^*_{i,1}, \sigma^*_{i,2})$ for $ID^*_i \in S^*$, where $\sigma_{i,2}$ is calculated in the usual way, and

$$\sigma_{i,0} = \mathcal{F}(\mathsf{ext}(T \cdot t'_i)) \oplus M^*_{\beta,i},\ \ \sigma_{i,1} = t'_i \cdot \prod_{j\in[1,l+1]} C_{n+j}.$$

  Finally, the challenger aggregates these signcryptions using the aggregation algorithm, and returns the result $\{\sigma^*, S^*\}$ to $\mathcal{A}$.
- ***Guess***: $\mathcal{A}$ outputs his guess $b' \in \{0,1\}$ for b.

If $b = 1$ then $\mathcal{A}$ played the proper security game. On the other hand, if $b = 0$, all information about the message M^*_b is lost. Therefore the advantage of $\mathcal{A}$ is exactly 0. As a result if $\mathcal{A}$ breaks the proper security game with a non-negligible advantage, then $\mathcal{B}$ has a non-negligible advantage in breaking the GGH (*l*+*n*)-MDDH assumption.

Acknowledgement

This work is partially supported by the National Natural Science Foundation of China (No.61272434), and the Natural Science Foundation of Shandong Province (No.ZR2013FQ021 and No.ZR2011FQ032).

References

1. Yuliang Zheng: Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). CRYPTO 1997: 165-179
2. Joonsang Baek, Ron Steinfeld, Yuliang Zheng: Formal Proofs for the Security of Signcryption. Public Key Cryptography 2002: 80-98
3. John Malone-Lee: Identity-Based Signcryption. IACR Cryptology ePrint Archive 2002: 98 (2002)
4. Benoit Libert, Jean-Jacques Quisquater: New identity based signcryption schemes from pairings. IACR Cryptology ePrint Archive 2003: 23 (2003)
5. Sherman S. M. Chow, Siu-Ming Yiu, Lucas Chi Kwong Hui, K. P. Chow: Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. ICISC 2003: 352-369
6. Xavier Boyen: Multipurpose Identity-Based Signcryption (A Swiss Army Knife for Identity-Based Cryptography). CRYPTO 2003: 383-399
7. Liqun Chen, John Malone-Lee: Improved Identity-Based Signcryption. Public Key Cryptography 2005: 362-379
8. Paulo S. L. M. Barreto, Benoit Libert, Noel McCullagh, Jean-Jacques Quisquater: Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. ASIACRYPT 2005: 515-532
9. S. Sharmila Deva Selvi, S. Sree Vivek, J. Shriram, S. Kalaivani, C. Pandu Rangan: Identity Based Aggregate Signcryption Schemes. INDOCRYPT 2009: 378-397
10. Alexander W. Dent: Aggregate Signcryption. IACR Cryptology ePrint Archive 2012: 200 (2012)
11. Jayaprakash Kar: Provably Secure Identity-Based Aggregate Signcryption Scheme in Random Oracles. IACR Cryptology ePrint Archive 2013: 37 (2013)
12. Dan Boneh, Alice Silverberg: Applications of Multilinear Forms to Cryptography. IACR Cryptology ePrint Archive 2002: 80 (2002)
13. Sanjam Garg, Craig Gentry, Shai Halevi: Candidate Multilinear Maps from Ideal Lattices. EUROCRYPT 2013: 1-17
14. Susan Hohenberger, Amit Sahai, Brent Waters: Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures. CRYPTO (1) 2013: 494-512
15. Hao Wang, Lei Wu, Zhihua Zheng, Yilei Wang: Identity-Based Key-Encapsulation Mechanism from Multilinear Maps. IACR Cryptology ePrint Archive 2013: 836 (2013)
16. Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, Christoph Striecks: Programmable Hash Functions in the Multilinear Setting. CRYPTO (1) 2013: 513-530

A Realization of Graded Encoding System

GGH's n-graded encoding system works as follows. (This is a whirlwind overview; see [13] for details.) The system uses three rings. First, it uses the ring of integers $\mathcal{O}$ of

the *m*-th cyclotomic field. This ring is typically represented as the ring of polynomials $\mathcal{O} = \mathbb{Z}[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is *m*-th cyclotomic polynomial, which has degree $N = \phi(m)$. Second, for some suitable integer modulus q, it uses the quotient ring $\mathcal{O}/(q) = \mathbb{Z}_q[x]/(\Phi_m(x))$. similar to the NTRU encryption scheme [27]. The encodings live in $\mathcal{O}/(q)$. Finally, it uses the quotient ring $R = \mathcal{O}/\mathcal{I}$, where $\mathcal{I} = \langle g \rangle$ is a principal ideal of $\mathcal{O}$ that is generated by g and where $|\mathcal{O}/\mathcal{I}|$ is a large prime. This is the ring "R" referred to above; elements of R are what is encoded.

What does a GGH encoding look like? For a fixed random $z \in \mathcal{O}/(q)$, an element of $S_i^{(\alpha)}$ - that is, a level-*i* encoding of $\alpha \in R$ - has the form $e/z^i \in \mathcal{O}/(q)$, where $e \in \mathcal{O}$ is a "small" representative of the coset $\alpha + \mathcal{I}$ (it has coefficients that are very small compared to q). To add encodings $e_1/z^i \in S_i^{(\alpha_1)}$ and $e_2/z^i \in S_i^{(\alpha_2)}$, just add them in $\mathcal{O}/(q)$ to obtain $(e_1 + e_2)/z^i$, which is in $S_i^{(\alpha_1+\alpha_2)}$ if $e_1 + e_2$ is "small". To mult encodings $e_1/z^{i_1} \in S_{i_1}^{(\alpha_1)}$ and $e_2/z^{i_2} \in S_{i_2}^{(\alpha_2)}$, just multiply them in $\mathcal{O}/(q)$ to obtain $(e_1 \cdot e_2)/z^{i_1+i_2}$, which is in $S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$ if $e_1 \cdot e_2$ is "small". This smallness condition limits the GGH encoding system to degree polynomial in the security parameter. Intuitively, dividing encodings does not "work", since the resulting denominator has a nontrivial term that is not z.

The GGH params allow everyone to generate encodings of random (known) values. The params include a level-1 encoding of 1 (from which one can generate encodings of 1 at other levels), and (for each $i \in [n]$) a sufficient number of level-*i* encodings of 0 to enable re-randomization. To encode (say at level-1), run samp(params) to sample a small element a from $\mathcal{O}$, e.g. according to a discrete Gaussian distribution. For a Gaussian with appropriate deviation, this will induce a statistically uniform distribution over the cosets of $\mathcal{I}$. Then, multiply a with the level-1 encoding of 1 to get a level-1 encoding u of $a \in R$. Finally, run reRand(params, $1, u$), which involves adding a random Gaussian linear combination of the level-1 encodings of 0, whose noisiness (i.e., numerator size) "drowns out" the initial encoding. The parameters for the GGH scheme can be instantiated such that the re-randomization procedure can be used for any pre-specified polynomial number of times.

To permit testing of whether a level-*n* encoding $u = e/z^n \in S_n$ encodes 0, GGH publishes a level-*n* zero-test parameter $\mathbf{p}_{zt} = hz^n/g$, where h is "somewhat small" and g is the generator of $\mathcal{I}$. The procedure isZero(params, $\mathbf{p}_{zt}$, u) simply computes $\mathbf{p}_{zt} \cdot u$ and tests whether its coefficients are small modulo q. If u encodes 0, then $e \in I$ and equals $g \cdot c$ for some (small) c, and thus $\mathbf{p}_{zt} \cdot u = h \cdot c$ has no denominator and is small modulo q. If u encodes something nonzero, $\mathbf{p}_{zt} \cdot u$ has g in the denominator and is not small modulo q. The ext(params, $\mathbf{p}_{zt}$, u) procedure works by applying a strong extractor to the most significant bits of $\mathbf{p}_{zt} \cdot u$. For any two $u_1, u_2 \in S_n^{(\alpha)}$, we have (subject to noise issues) $u_1 - u_2 \in S_n^{(0)}$, which implies $\mathbf{p}_{zt}(u_1 - u_2)$ is small, and hence $\mathbf{p}_{zt} \cdot u_1$ and $\mathbf{p}_{zt} \cdot u_2$ have the same most significant bits (for an overwhelming fraction of α's).

Garg et al. provide an extensive cryptanalysis of the encoding system, which we will not review here. We remark that the underlying assumptions are stronger, but related to, the hardness assumption underlying the NTRU encryption scheme: that it is hard to distinguish a uniformly random element from $\mathcal{O}/(q)$ from a ratio of "small" elements

i.e., an element $u/v \in \mathcal{O}/(q)$ where $u, v \in \mathcal{O}/(q)$ both have coefficients that are on the order of (say) q^{ϵ} for small constant ϵ.