# Identity-Based Key-Encapsulation Mechanism from Multilinear Maps

Hao Wang[1,2,3⋆], Lei Wu[1,2], Zhihua Zheng[1,2], and Yilei Wang[4]

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

**Abstract.** We construct an Identity-Based Key Encapsulation Mechanism (IB-KEM) in a generic "leveled" multilinear map setting and prove its security under multilinear decisional Diffie-Hellmanin assumption in the selective-ID model. Then, we make our IB-KEM translated to the GGH framework, which defined an "approximate" version of a multilinear group family from ideal lattices, and modify our proof of security to use the GGH graded algebras analogue of multilinear maps.

## 1 Introduction

An Identity Based Encryption (IBE) system [1] is a public key system where the public key can be an arbitrary string such as an email address. A central authority, called a Private Key Generator (PKG), uses a master key to issue private keys to identities that request them. Instead of providing the full functionality of an IBE scheme, in many applications it is sufficient to let sender and receiver agree on a common random session key. This can be accomplished with an Identity Based Key Encapsulation Mechanism (IB-KEM) as formalized in [2]. Any IB-KEM can be updated to a full IBE scheme by adding a symmetric encryption scheme with appropriate security properties.

There are currently three classes of IBE (IB-KEM) systems: (1) based on groups with a bilinear map [3–7] (to name a few), (2) based on quadratic residuosity modulo a composite [8–10], and (3) based on hard problems on lattices [11, 12].

In this paper we present an IB-KEM construction based on groups with a *multilinear map* [13]. We present our IB-KEM in a generic "leveled" multilinear map setting and prove its security in the selective-ID model. Then, we make our IB-KEM translated to the GGH framework [14], which defined an "approximate" version of a multilinear group family from ideal lattices.

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

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

## 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 [14].

For generic, leveled multilinear maps. We assume the existence of a group generator $\mathcal{G}$, which takes as input a security parameter $1^\lambda$ 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) [14] defined an "approximate" version of a multilinear group family, which they call a *graded encoding system*. As a starting point, they view $g_i^\alpha$ in a multilinear group family as simply an *encoding* of $\alpha$ 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 $\alpha$". 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 $\lambda$ and integer $k$. The procedure outputs ($\mathsf{params}$, $\mathbf{p}_{zt}$), where $\mathsf{params}$ 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 $\alpha$ 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 $\alpha$.

**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 isZero(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 ext(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 $u_1, u_2 \in S_k^{(\alpha)}$, ext(params, $\mathbf{p}_{zt}$, $u_1$) = ext(params, $\mathbf{p}_{zt}$, $u_2$),
- (b) The distribution $\{$ext(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 Decisional Diffie-Hellman: *k*-MDDH)** *The n-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 $\mathbb{G}_n$-element with better than negligible advantage (in security parameter $\lambda$).*

**Assumption 2 (GGH analogue of *k*-MDDH: GGH *k*-MDDH)** *The GGH k-Multilinear Decisional Diffie-Hellman (k-MDDH) problem states the following: A challenger runs* $\mathsf{InstGen}(1^\lambda, 1^k)$ *to obtain (*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* $\mathsf{params}$, $\mathbf{p}_{zt}$, $\mathsf{enc}(1, c_1)$, ..., $\mathsf{enc}(1, c_{k+1})$ *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* $\lambda$*).*

## 3 Definitions for Identity-Based Key Encapsulation Mechanism

### 3.1 Identity-Based Key Encapsulation Mechanism

An IB-KEM consists of four PPT algorithms as follows:

– **Setup**($1^\lambda$): take as input a security parameter $\lambda$, output the public parameters *PP* and the master secret key *MSK*. *PP* may be used as an implicit input for algorithms **KeyGen**, **Encap**, **Decap**. Let $\mathcal{I}$ be the identity space, $\mathcal{C}$ be the ciphertext space, and $\mathcal{K}$ be the DEM key space.
– **KeyGen**(*MSK*, *ID*): take as input *PP*, *MSK* and an identity $ID \in \mathcal{I}$, output a private key $SK_{ID}$ of *ID*.
– **Encap**(*PP*, *ID*): take as input *PP* and an identity $ID \in \mathcal{I}$, output a ciphertext $C$ and a DEM key $K \in \mathcal{K}$.
– **Decap**($SK_{ID}$, $C$): take as input a private key $SK_{ID}$ for identity $ID$ and a ciphertext $C \in \mathcal{C}$, output a DEM key $K \in \mathcal{K}$ or a special reject symbol $\perp$ (which is not in $\mathcal{K}$) indicating that $C$ is not consistent under $ID$.

**Correctness** For correctness, we require that for any identities $ID \in \mathcal{I}$, and any $(C, K)\leftarrow$**Encap**(*PP*, *ID*), **Decap**(**KeyGen**(*MSK*, *ID*), $C$) = $K$ holds overwhelmingly, where the probability is taken over the choice of (*PP*, *MSK*)$\leftarrow$**Setup**($1^\lambda$), and the random coins of all the algorithms in the expression above.

### 3.2 Security Game

We define IB-KEM security under a selective-identity attack using the following game between a challenger and an adversary $\mathcal{A}$:

– ***Init***: The adversary outputs an identity $ID^*$ where it wishes to be challenged.
– ***Setup***: The challenger runs the **Setup** algorithm giving it the security parameter as input. It gives $\mathcal{A}$ the resulting public parameters *PP*.
– ***Phase 1***: In this phase the adversary $\mathcal{A}$ can adaptively ask for secret keys for any identities except $ID^*$. For each queried identity $ID$, the challenger calls **KeyGen**(*MSK*, $ID$) $\rightarrow SK_{ID}$ and sends $SK_{ID}$ to the adversary. (The restriction that has to be satisfied for each query is that none of the queried identity is identical to $ID^*$.)
– ***Challenge***: The challenger samples $K_0^*\leftarrow\mathcal{K}$, and computes $(C^*, K_1^*)\leftarrow$**Encap**(*PP*, *ID**)). Then, it flips a random coin $b \in \{0, 1\}$ and sends ($C^*$, $K_b^*$) to $\mathcal{A}$.
– ***Phase 2***: This the same as query phase 1.
– ***Guess***: The adversary outputs his guess $b' \in \{0, 1\}$ for $b$.

**Definition 1.** *A IB-KEM scheme is selectively secure under chosen plaintext attack (IND-sID-CPA) if all PPT adversaries have at most a negligible advantage in $\lambda$ in the above security game, where the advantage of an adversary is defined as $Adv = Pr[b' = b] - 1/2$.*

# 4 IB-KEM in Generic "Leveled" Multilinear Map Setting

In this section, we give our identity-based key encapsulation mechanism in a generic "leveled" multilinear map setting, using the construction of full domain hash function introduced by Hohenberger, Sahai and Waters in [16]. Then, we prove its security under $n$-MDDH assumption.

## 4.1 Generic Multlinear Construction

**Setup**($1^\lambda$, $n$): 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 $n$ of identities. It first runs $\mathcal{G}(1^\lambda, n)$ and outputs a sequence of groups $\overrightarrow{\mathbb{G}} = (\mathbb{G}_1, ..., \mathbb{G}_n)$ of prime order $p$, with canonical generators $g_1, ..., g_n$, where we let $g = g_1$.

Next, it 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 the function $H(ID) : \{0, 1\}^n \to \mathbb{G}_n$. Let $id_1, ..., id_n$ as the bits of $ID$. It is computed iteratively as

$$H_1(ID) = B_{1,id_1}, H_i(ID) = e(H_{i-1}(ID), B_{i,id_i}) \ \ for\ i \in [2, n]$$

It defines $H(ID) = H_n(ID)$, and sets a randomness extractor, $s \leftarrow \mathsf{ext}(S)$, where $s \in \{0, 1\}^\lambda$, $S \in \mathbb{G}_n$.

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

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

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 $SK_{ID} = g_{n-1}^{\prod_{i\in[1,n]} b_{i,id_i}} \in \mathbb{G}_{n-1}$.

**Encap**(*PP*, $ID = (id_1, ..., id_n)$) : The encapsulation algorithm chooses $t \in \mathbb{Z}_p$ randomly and outputs

$$C = g^t, \ \ K = \mathsf{ext}(H(ID)^t).$$

**Decap**($SK_{ID}$, $C$): The decapsulation algorithm computes that

$$K = \mathsf{ext}(e(SK_{ID}, C)).$$

**Correctness**

$$\begin{aligned} H(ID) &= H_n(ID) \\ &= e(H_{n-1}(ID), B_{n,id_n}) \\ &= e(e(H_{n-2}(ID), B_{n-1,id_{n-1}}), B_{n,id_n}) \\ &\ldots\ldots \\ &= e(B_{1,id_1}, ..., B_{n,id_n}) \\ &= e(g_1^{b_{1,id_1}}, ..., g_1^{b_{n,id_n}}) \\ &= g_n^{\prod_{i\in[1,n]} b_{i,id_i}} \end{aligned}$$

$$\begin{aligned} e(SK_{ID}, C) &= e(g_{n-1}^{\prod_{i\in[1,n]} b_{i,id_i}}, g^t) \\ &= g_n^{\prod_{i\in[1,n]} b_{i,id_i} \cdot t} \end{aligned}$$

Therefore, $K = \mathsf{ext}(e(SK_{ID}, C)) = \mathsf{ext}(H(ID)^t)$.

### 4.2 Security

We will prove the following theorem regarding the selective security of our IB-KEM:

**Theorem 1.** *If the n-MDDH assumption holds then our scheme is selectively secure under chosen plaintext attack (IND-sID-CPA).*

*Proof.* Suppose $\mathcal{A}$ has a non negligible advantage in attacking the IB-KEM. We build an algorithm $\mathcal{B}$ that breaks the *n*-MDDH assumption. Algorithm $\mathcal{B}$ is given as input $g = g_1$, $g^{c_1}$, ..., $g^{c_{n+1}}$, $T$ , where $T$ is identical to $g_n^{\prod_{j\in[1,n+1]} c_j}$ or uniform and independent in $\mathbb{G}_n$. Algorithm $\mathcal{B}$'s goal is to output 1 if $T = g_n^{\prod_{j\in[1,n+1]} c_j}$ and 0 otherwise.

- ***Init***: $\mathcal{A}$ outputs an identity $ID^* = (id_1^*, ..., id_n^*)$, where it wishes to be challenged, the $id_i^*$ is the *i*-th bit of $ID^*$.
- ***Setup***: $\mathcal{B}$ 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 $\mathsf{ext} : \mathbb{G}_n \to \{0, 1\}^l$, and sends $(B_{1,0}, B_{1,1}), ..., (B_{n,0}, B_{n,1})$, $\mathsf{ext}$ as well as the group sequence description to $\mathcal{A}$.
- ***Phase 1 & 2***: $\mathcal{B}$ has to produce secret keys for any identities $ID_i \neq ID^*$ requested by $\mathcal{A}$. In both phases the treatment is the same. We describe here the way $\mathcal{B}$ works in order to create a key for $ID_i = (id_{i,1}, ..., id_{i,n})$. Science $ID_i \neq ID^*$, there exists at least one bit $id_{i,j} \neq id_j^*$, where $j \in [1, n]$. $\mathcal{B}$ can calculate the secret key:

$$SK_{ID_i} = e(B_{1,id_{i,1}}, B_{2,id_{i,2}}, ..., B_{j-1,id_{i,j-1}}, B_{j+1,id_{i,j+1}}, ..., B_{n,id_{i,n}})^{b_j}$$

- ***Challenge***: $\mathcal{B}$ constructs $C^* = g^{c_{n+1}}$, $K_0^* \leftarrow \{0, 1\}^\lambda$, $K_1^* = \mathsf{ext}(T)$, and flips a random coin $b \in \{0, 1\}$. Then, $\mathcal{B}$ sends $(C^*, K_b^*)$ 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 $K_b^*$ 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 *n*-MDDH assumption.

## 5 IB-KEM in the GGH Framework

In this section, we show how to modify our ID-based construction to use the GGH [14] 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 $\mathsf{params}$ 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 $\mathsf{params}$ throughout the scheme, so there is no cause for confusion. For further details on the GGH framework, please refer to [14]. The realization method of GGH's graded encoding system is included in Appendix A.

### 5.1 Construction in the GGH Framework

**Setup**($1^\lambda$, $n$): 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 $n$ of identities. It then runs ($\mathsf{params}$, $\mathbf{p}_{zt}$)←$\mathsf{InstGen}(1^\lambda, 1^n)$. Recall that $\mathsf{params}$ will be implicitly given as input to all GGH-related algorithms below.

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

These will be used to compute a function $H$ mapping $n$ bit strings to level $n$ encodings. Let $id_1$, ..., $id_n$ as the bits of $ID$. It is computed iteratively as

$$H_1(ID) = B_{1,id_1}, H_i(ID) = \mathsf{mult}(H_{i-1}(ID), B_{i,id_i}) \;\; for \; i \in [2, n]$$

It defines $H(ID) = \mathsf{reRand}(n, H_n(ID))$.

The public parameters, *PP*, consist of the $\mathsf{params}$, ($B_{1,0}$, $B_{1,1}$), ..., ($B_{n,0}$, $B_{n,1}$) and extractor $\mathsf{ext}$.

Note that $\mathsf{params}$ includes a level 1 encoding of 1, which we denote as $g$.

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 $SK_{ID} = \mathsf{reRand}(n-1, \mathsf{enc}(n-1, \prod_{i\in[1,n]} b_{i,id_i}))$.
**Encap**(*PP*, $ID = (id_1, ..., id_n)$) : The encapsulation algorithm chooses random encodings $t = \mathsf{samp}()$ and outputs

$$C = \mathsf{enc}(1, t), \;\; K = \mathsf{ext}(\mathbf{p}_{zt}, \mathsf{mult}(H(ID), t)).$$

**Decap**($SK_{ID}$, $C$) : The decapsulation algorithm computes that

$$K = \mathsf{ext}(\mathbf{p}_{zt}, \mathsf{mult}(SK_{ID}, C))$$

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

### 5.2 Proof of Security for IB-KEM in the GGH framework

We now describe how to modify our proof of security for our IB-KEM to use the GGH [14] 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 on the GGH framework, please refer to [14].

*Proof*. Suppose $\mathcal{A}$ has a non negligible advantage in attacking the IB-KEM in the G-GH Framework. We build an algorithm $\mathcal{B}$ that breaks the GGH $n$-MDDH assumption. Algorithm $\mathcal{B}$ is given as input params, $\mathbf{p}_{zt}$, $\mathsf{enc}(1, c_1)$, ..., $\mathsf{enc}(1, c_{k+1})$ and a level-$k$ encoding $T$. Algorithm $\mathcal{B}$'s goal is to output 1 if $\mathsf{isZero}(\boldsymbol{p}_{zt}, \mathsf{reRand}(T) - \mathsf{reRand}(\mathsf{enc}(\mathsf{params}, k, \prod_{j\in[1,k+1]} c_j))) = 1$ and 0 otherwise.

- ***Init***: $\mathcal{A}$ outputs an identity $ID^* = (id_1^*, ..., id_n^*)$, where it wishes to be challenged, the $id_i^*$ is the $i$-th bit of $ID^*$.
- ***Setup***: $\mathcal{B}$ chooses random encodings $b_i = \mathsf{samp}()$ for $i \in [1, n]$ and sets ($B_{i,id_i^*}$ = $\mathsf{enc}(1, c_i)$, $B_{i,(1-id_i^*)}$ = $\mathsf{enc}(1, b_i)$) for $i \in [1, n]$. Then, it sends $(B_{1,0}, B_{1,1})$, ..., $(B_{n,0}, B_{n,1})$, ext as well as params to $\mathcal{A}$.
- ***Phase 1 & 2***: $\mathcal{B}$ has to produce secret keys for any identities $ID_i \neq ID^*$ requested by $\mathcal{A}$. In both phases the treatment is the same. We describe here the way $\mathcal{B}$ works in order to create a key for $ID_i = (id_{i,1}, ..., id_{i,n})$. Science $ID_i \neq ID^*$, there exists at least one bit $id_{i,j} \neq id_j^*$, where $j \in [1, n]$. $\mathcal{B}$ can calculate the secret key:

$$SK_{ID_i} = \mathsf{reRand}(n-1, \prod_{k\in[1,j-1]\bigcup[j+1,n]} B_{k,id_{i,k}} \cdot b_j)$$

- ***Challenge***: $\mathcal{B}$ constructs $C^*$=$\mathsf{enc}(1, c_{k+1})$, $K_0^* \leftarrow \{0,1\}^\lambda$, $K_1^* = \mathsf{ext}(T)$, and flips a random coin $b \in \{0, 1\}$. Then, $\mathcal{B}$ sends $(C^*, K_b^*)$ 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 $K_b^*$ 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 $n$-MDDH assumption.

## References

1. Adi Shamir: Identity-Based Cryptosystems and Signature Schemes. CRYPTO 1984: 47-53
2. Kamel Bentahar, Pooya Farshim, John Malone-Lee, Nigel P. Smart: Generic Constructions of Identity-Based and Certificateless KEMs. J. Cryptology 21(2): 178-199 (2008)
3. Dan Boneh, Matthew K. Franklin: Identity-Based Encryption from the Weil Pairing. CRYPTO 2001: 213-229
4. Dan Boneh, Xavier Boyen: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. EUROCRYPT 2004: 223-238
5. Dan Boneh, Xavier Boyen: Secure Identity Based Encryption Without Random Oracles. CRYPTO 2004: 443-459
6. Brent Waters: Efficient Identity-Based Encryption Without Random Oracles. EUROCRYPT 2005: 114-127

7. Craig Gentry: Practical Identity-Based Encryption Without Random Oracles. EUROCRYPT 2006: 445-464
8. Clifford Cocks: An Identity Based Encryption Scheme Based on Quadratic Residues. IMA Int. Conf. 2001: 360-363
9. Dan Boneh, Craig Gentry, Michael Hamburg: Space-Efficient Identity Based Encryption Without Pairings. FOCS 2007: 647-657
10. Giovanni Di Crescenzo, Vishal Saraswat: Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. INDOCRYPT 2007: 282-296
11. Craig Gentry, Chris Peikert, Vinod Vaikuntanathan: Trapdoors for hard lattices and new cryptographic constructions. STOC 2008: 197-206
12. Shweta Agrawal, Dan Boneh, Xavier Boyen: Efficient Lattice (H)IBE in the Standard Model. EUROCRYPT 2010: 553-572
13. Dan Boneh, Alice Silverberg: Applications of Multilinear Forms to Cryptography. IACR Cryptology ePrint Archive 2002: 80 (2002)
14. Sanjam Garg, Craig Gentry, Shai Halevi: Candidate Multilinear Maps from Ideal Lattices. EUROCRYPT 2013: 1-17
15. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, Brent Waters: Attribute-Based Encryption for Circuits from Multilinear Maps. CRYPTO (2) 2013: 479-499
16. Susan Hohenberger, Amit Sahai, Brent Waters: Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures. CRYPTO (1) 2013: 494-512

## A Realization of Graded Encoding System

GGH's n-graded encoding system works as follows. (This is a whirlwind overview; see [14] 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 $\mathsf{mult}$ 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 $\mathsf{params}$ allow everyone to generate encodings of random (known) values. The $\mathsf{params}$ 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 $\mathsf{samp}(\mathsf{params})$ 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 $\mathsf{reRand}(\mathsf{params}, 1, u)$, 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 $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u)$ 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 $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ 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 $\alpha$'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) $q^\epsilon$ for small constant $\epsilon$.