Copyright ©The Institute of Electronics,
Information and Communication Engineers

SCIS 2014 The 31st Symposium on
Cryptography and Information Security
kagoshima, Japan, Jan. 21 - 24, 2014
The Institute of Electronics,
Information and Communication Engineers

Multilinear Maps on LWE

Ryo Hiromasa* Masayuki Abe† Tatsuaki Okamoto†

Abstract— Bilinear elliptic curve pairings are widely used in cryptography. Extending them to multilinear analogues is a long standing open problem. Two plausible candidates of multilinear maps have been already proposed by Garg, Gentry and Halevi, and by Coron, Lepoint and Tibouchi. They implicitly rely on the hardness of the short integer solution problem on ideal lattices, and the error-free approximate greatest common divisor problem, respectively. We construct new multilinear maps from fully homomorphic encryption (FHE) proposed by Gentry, Sahai, and Waters. The FHE is based on the learning with errors assumption (that is a bit more standard than the above hardnesses), and so are our maps.

Keywords: fully homomorphic encryption, multilinear maps, learning with errors

1 Introduction

1.1 Background

Bilinear elliptic curve pairings are widely used in cryptography. Extending them to multilinear maps is a long standing open problem. Candidate constructions were recently proposed in the innovational work by Garg, Gentry and Halevi [GGH13a]. After that, Coron, Lepoint and Tibouchi discovered an alternative construction [CLT13].

The essential difference with the bilinear maps is about the encoding $\alpha \cdot g$ of an element α. While the bilinear maps encode elements deterministically, the encoding of the multilinear maps is *noisy*. The multilinear maps is bounded to a polynomial degree. The encoding noise grows with the degree, and the noise overwhelms the signal for very high degree. This is like for ciphertexts in somewhat homomorphic encryptions. The security of their construction is based on the Diffie-Hellman-like assumption, which has rigorous scrutinies but have not been known to be reduced to established assumptions.

We introduce a brief overview of the GGH construction [GGH13a]. The construction works in the polynomial ring R. It generates a ring element $\mathbf{g} \in R$, and has the principal prime ideal $\mathfrak{p} = \langle \mathbf{g} \rangle \subseteq R$. In addition, it has an integer modulus $q \in \mathbb{Z}$ and a secret ring element $\mathbf{z} \in R$. The term $\alpha \cdot g$ in the discrete-log system is viewed as the encoding of the plaintext element α. In the GGH construction, the plaintext is played by the elements in the quotient ring $R/\mathfrak{p}$. To encode the plaintext, the construction divide it by $\mathbf{z}$. The construction provides many levels of encodings: the level-i encoding of a coset $\mathbf{e}_\mathfrak{p} = \mathbf{e} + \mathfrak{p}$ is formed by $\mathbf{c}/\mathbf{z}^i \mod q$ where $\mathbf{c} \in \mathbf{e}_\mathfrak{p}$ is short. One can add and multiply the encodings as long as the size of the numerator is shorter than q. In particular, the product of κ level-1 encodings yields a level-κ encoding. To test if two encodings at the maximum level κ have the same coset, the GGH construction defines a certain parameter, called the *zero-testing parameter*. The zero-testing parameter, which is included in public parameters, is the element $\mathbf{p}_{zt} = \mathbf{h} \cdot \mathbf{z}^\kappa/\mathbf{g} \mod q$ for a small $\mathbf{h} \in R$. Multiplying the encoding of 0 by $\mathbf{p}_{zt}$ results in a small element, while multiplying the encoding of non-zero element by $\mathbf{p}_{zt}$ results in a large element. Therefore one can distinguish if two encodings encodes the same coset.

Multilinear maps have many applications, most notably *program obfuscation* [BGK+13, BR13a, BR13b, GGH+13b, PTS13]. The goal of program obfuscation is to make a computer program unintelligible for hiding its implementation details, while preserving its functionality.

1.2 Our Results

We construct new multilinear maps from fully homomorphic encryption (FHE) proposed by Gentry, Sahai, and Waters [GSW13]. The FHE is based on the *learning with errors assumption*, and so are our maps. This is a bit more standard than what the others assume.

- The GGH construction relies on the conjectured hardness of the *short integer solution* (SIS) problem on ideal lattices. In a nutshell, the SIS problem is to find a short vector in a lattice. Given parameters and a challenge in the graded Diffie-Hellman assumption (described in Section 2.4) for the GGH construction, an adversary can compute a (not short) basis for the principal ideal $\mathfrak{p} = \langle \mathbf{g} \rangle$. Solving the SIS problem implies that one can obtain a short element in $\mathfrak{p}$. With the help of the short element, as described in [GGH13a], one can verify the challenge in the GDDH assumption.

- In the CLT construction [CLT13], level-0 encodings are ciphertexts similar to the batch variant of FHE based on integers [CCK+13]. Its public parameters include the set of level-0 encodings that encode random secret plaintexts. The construction requires that the level-0 encodings do not leak any information about the secrets. For giving this assurance, the CLT con-

* Kyoto University
† NTT and Kyoto University

struction implicitly relies on the hardness of the *error-free approximate common divisor* problem.

1.3 Our Techniques

In [GSW13], Gentry et al. proposed a very simple FHE (In the following, we call it GSW-FHE.), which is easy to write, does not have additional procedures to reduce noise, and so is asymptotically faster than other FHE (based on the standard LWE assumption). GSW-FHE has ciphertexts formed by matrices $\mathbf{C} \in \mathbb{Z}_q^{N\times N}$ for $N \in \mathbb{N}$ and a modulus $q \in \mathbb{Z}$. Let $\mathbf{t} \in \mathbb{Z}_q^N$ be a secret key. $\mathbf{C}$ and $\mathbf{t}$ satisfy the equation $\mathbf{Ct} = \alpha \cdot \mathbf{t} + \mathbf{e} \mod q$ for a plaintext $\alpha \in \mathbb{Z}_q$ and a noise vector $\mathbf{e} \in \mathbb{Z}_q^N$. One can decrypt α correctly as long as $\mathbf{e}$ is small. Let $\mathbf{C}_1, \mathbf{C}_2$ be two ciphertexts with plaintexts α_1, α_2 and noise vectors $\mathbf{e}_1, \mathbf{e}_2$. For addition, we have

$$(\mathbf{C}_1 + \mathbf{C}_2)\mathbf{t} = (\alpha_1 + \alpha_2) \cdot \mathbf{t} + (\mathbf{e}_1 + \mathbf{e}_2) \mod q.$$

For multiplication, we have

$$(\mathbf{C}_1\mathbf{C}_2)\mathbf{t} = \mathbf{C}_1(\alpha_2 \cdot \mathbf{t} + \mathbf{e}_2) = \alpha_1\alpha_2 \cdot \mathbf{t} + (\alpha_2\mathbf{e}_1 + \mathbf{C}_1\mathbf{e}_2) \mod q.$$

In the above two equations, the decryption is correct as long as the final noise terms are small.

Roughly speaking, our encodings are formed by the GSW-FHE ciphertext divided by an integer $z \in \mathbb{Z}_q$. A level-0 encoding is just a GSW-FHE ciphertext $\mathbf{C}$, and a level-i encoding is $\mathbf{C}/z^i$. It is easy to see that one can add and multiply the encodings. We define the permuted secret key $\mathbf{t}_{perm}$ and permuted public key $\mathbf{B}_{perm}$ such that for a noise vector $\mathbf{e}$ and a ciphertext $\mathbf{C}$ that encrypts α by $\mathbf{B}_{perm}$, we have

$$\mathbf{Ct}_{perm} = \alpha \cdot \mathbf{t}_{perm} + \mathbf{e} \mod q.$$

This is similar to the equation related to the normal secret key, but the decryption is jammed by the permutation. We define the zero-testing parameter as $\mathbf{p}_{zt} = z^k \cdot \mathbf{t}_{perm} \mod q$. Given a level-$\kappa$ (that is the maximum level) encoding $\mathbf{U} = \mathbf{C}/z^\kappa$ that encodes a plaintext α and has a noise vector $\mathbf{e}$, we have

$$\mathbf{w} = \mathbf{Up}_{zt} = \frac{\mathbf{U}}{z^\kappa} \cdot z^\kappa \cdot \mathbf{t}_{perm} = \mathbf{Ut}_{perm} = \alpha \cdot \mathbf{t}_{perm} + \mathbf{e} \mod q.$$

We can see that $\mathbf{w}$ is small if $\alpha = 0$, and large otherwise. Therefore, by subtracting two encodings and multiplying the result by the zero-testing parameter, one can distinguish if they contain the same plaintext.

2 Preliminaries

2.1 Notations

We denote the set of integers by $\mathbb{Z}$ and the set of natural numbers by $\mathbb{N}$. Let $\mathbb{G}$ be some group and $\mathcal{P}$ be some probability distribution, then we use $a \xleftarrow{U} \mathbb{G}$ to denote that a is chosen from $\mathbb{G}$ uniformly at random, and use $b \xleftarrow{R} \mathcal{P}$ to denote that b is chosen along $\mathcal{P}$. We take all logarithms to the base 2, unless otherwise noted. We let $\mathsf{negl}(\lambda)$ denote a negligible function of λ. We use standard big-O notation to classify the growth of functions. For two randam variables X, Y, $\Delta(X, Y) := 1/2|Pr[X = x] - Pr[Y = y]|$ represents the statistical distance between X and Y, and for a security parameter λ the notation $X \overset{s}{\approx} Y$ refer that $\Delta(X, Y) \leq 2^{-\lambda}$.

We assume that vectors are in column form and are written by using bold lower-case letters, e.g., $\mathbf{x}$. The ith element of a vector is denoted by x_i. We let the length of vectors be the l_∞ norm of the vectors and denote it by $\| \mathbf{x} \|$. The inner product between two vectors is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. Matrices are written by using bold capital letters, e.g., $\mathbf{X}$, and the ith column vector of a matrix is denoted by $\mathbf{x}_i$. For a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, the notation $\mathbf{X}^T \in \mathbb{R}^{n\times m}$ denotes the transpose of $\mathbf{X}$. For two matrices $\mathbf{A} \in \mathbb{R}^{m\times n_1}$ and $\mathbf{B}^{m\times n_2}$, $[\mathbf{A} \parallel \mathbf{B}] \in \mathbb{R}^{m\times(n_1+n_2)}$ denotes the concatenation of $\mathbf{A}$ with $\mathbf{B}$. When we refer to the $n \times n$ identity matrix, we denote it by $\mathbf{I}_n$.

2.2 Graded Encoding System

Garg et al. [GGH13a] construct the *graded encoding system* to achieve multilinear maps. The formal definition of the system is like below.

Definition 2.1 (κ-Graded Encoding System)**.** *A κ-graded encoding system for a ring R is a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subseteq \{0,1\}^* : 0 \leq i \leq \kappa, \alpha \in R\}$ with the following properties:*

1. *For every $i \in \mathbb{N}$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint.*
2. *There are associative binary operations '+' and '−' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every index $i \in [\kappa]$, and every $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that*
$$u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)} \text{ and } u_1 - u_2 \in S_i^{(\alpha_1-\alpha_2)}$$
where $\alpha_1+\alpha_2$ and $\alpha_1-\alpha_2$ are addition and subtraction in R.
3. *There is an associative binary operation '$\times$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every index i_1, i_2 with $0 \leq i_1, i_2 \leq \kappa$, and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$, it holds that*
$$u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1\cdot\alpha_2)}$$
where $\alpha_1 \cdot \alpha_2$ is multiplication in R.

2.3 Procedures for Manipulating Encodings

Garg et al. gave the instantiation to the above encoding system. Unlike the other encoding systems [GGH13a, CLT13], our instantiation allows users to sample a level-0 encoding for any plaintext element, so we add the interface to the sampling algorithm.

Instance generation. Given as input parameters λ and κ, the instance generation algorithm $\mathsf{InstGen}$ outputs parameters params for graded encoding systems and a zero-testing parameter $\mathbf{p}_{zt}$.

Sampling level-zero encodings. The sampling algorithm Samp takes as input an plaintext element $\alpha \in R$, and outputs a level-0 encoding $\mathbf{U} \in S_0^{(\alpha)}$ for α.

Encoding. The algorithm Encode takes as input a level i and a level-0 encoding $\mathbf{C} \in S_0^{(\alpha)}$, and outputs a level-$i$ encoding $\mathbf{U}_i \in S_i^{(\alpha)}$ for the same α as $\mathbf{C}$.

Addition and negation. Given as input two encodings $\mathbf{U}_1 \in S_i^{(\alpha_1)}$ and $\mathbf{U}_2 \in S_i^{(\alpha_2)}$, it holds that $\mathsf{Add}(\mathsf{params}, \mathbf{U}_1, \mathbf{U}_2) \in S_i^{(\alpha_1+\alpha_2)}$ and $\mathsf{Neg}(\mathsf{params}, \mathbf{U}_1, \mathbf{U}_2) \in S_i^{(\alpha_1-\alpha_2)}$.

Multiplication. Given as input two encodings $\mathbf{U}_1 \in S_{i_1}^{(\alpha_1)}, \mathbf{U}_2 \in S_{i_2}^{(\alpha_2)}$ such that $i_1 + i_2 \leq \kappa$, we have $\mathsf{Mult}(\mathsf{params}, \mathbf{U}_1, \mathbf{U}_2) \in S_{i_1+i_2}^{(\alpha_1\alpha_2)}$.

Zero-testing. The zero-testing procedure isZero takes as input a level-κ encoding $\mathbf{U}_\kappa \in S_\kappa^{(\alpha)}$ and a zero-testing parameter $\mathbf{p}_{zt}$, and outputs 1 if $\mathbf{U}_\kappa \in S_\kappa^{(0)}$ and 0 otherwise.

Extraction. Given as input a zero-testing parameter $\mathbf{p}_{zt}$ and a level-κ encoding $\mathbf{U} \in S_\kappa^{(\alpha)}$ for $\alpha \in R$, the algorithm $\mathsf{Extract}$ outputs a string $s \in \{0,1\}^\lambda$ that has the following properties:

1. For any $\alpha \in R$, let $\mathbf{U}, \mathbf{U}' \in S_\kappa^{(\alpha)}$.

$$\mathsf{Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}) = \mathsf{Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}').$$

2. A distribution

$$\{\mathsf{Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}) \in \{0,1\}^\lambda \mid \alpha \in R, \mathbf{U} \in S_\kappa^{(\alpha)}\}$$

is nearly close to uniform over $\{0,1\}^\lambda$.

2.4 Hardness assumptions

Here we introduce assumptions that we mention in this paper.

The Learning with Errors Assumption. The *learning with errors (LWE) assumption* was introduced by Regev [Reg05].

Definition 2.2 (LWE)**.** *For a security parameter λ, let $n := n(\lambda)$ be an integer dimension, let $q := q(\lambda) \geq 2$ be an integer modulus, and let $\chi := \chi(\lambda)$ be a discrete Gaussian distribution over $\mathbb{Z}$. The LWE assumption is that it is difficult to distinguish the following two distributions: In the first distribution, a tuple $(\mathbf{a}_i, b_i)$ is sampled from uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. In the second distribution, $\mathbf{s} \xleftarrow{U} \mathbb{Z}_q^n$, and then a tuple $(\mathbf{a}_i, b_i)$ is sampled by sampling $\mathbf{a}_i \xleftarrow{U} \mathbb{Z}_q^n$, $e_i \xleftarrow{R} \chi$, and setting $b_i := \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod q$.*

The Graded Decisional Diffie Hellman Assumption. The *graded decisional Diffie-Hellman (GDDH) assumption* is a multilinear analogue of the bilinear decisional Diffie-Hellman (BDDH) assumption.

We consider the following process.

1. $(\mathsf{params}, \mathbf{p}_{zt}) \xleftarrow{R} \mathsf{InstGen}(1^\lambda, 1^\kappa)$.
2. $\mathbf{C}_i \xleftarrow{R} \mathsf{Samp}(\mathsf{params})$ for $i = 1, \ldots, \kappa + 1$.
3. $\mathbf{U}_i \xleftarrow{R} \mathsf{Encode}(\mathsf{params}, 1, \mathbf{C}_i)$ for $i = 1, \ldots, \kappa + 1$.
4. $\hat{\mathbf{C}} \xleftarrow{R} \mathsf{Samp}(\mathsf{params})$.
5. $\tilde{\mathbf{U}} = \mathbf{C}_{\kappa+1} \times \prod_{i=1}^{\kappa} \mathbf{U}_i$.
6. $\hat{\mathbf{U}} = \hat{\mathbf{C}} \times \prod_{i=1}^{\kappa} \mathbf{U}_i$.

Definition 2.3 (GDDH)**.** *The GDDH assumption is to distinguish the following two distributions:*

$$\mathcal{D}_{GDDH} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{\mathbf{U}_i\}_{i\in[\kappa+1]}, \tilde{\mathbf{U}})\},$$
$$\mathcal{D}_{RAND} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{\mathbf{U}_i\}_{i\in[\kappa+1]}, \hat{\mathbf{U}})\}.$$

The GDDH assumption states that for any efficient adversary $\mathcal{A}$ and a security parameter λ,

$$|Pr[\mathcal{A}(\mathcal{D}_{GDDH}) \to 1] - Pr[\mathcal{A}(\mathcal{D}_{RAND}) \to 1]| < \mathsf{negl}(\lambda).$$

2.5 Min-Entropy, Leftover Hash Lemma, Strong Randomness Extractor

Definition 2.4 (Min-Entropy)**.** *A distribution $\mathcal{D}$ has min-entropy, denoted by $H_\infty(\mathcal{D}) \geq k$, if*

$$\max_{d \xleftarrow{R} \mathcal{D}} Pr[\mathcal{D} = d] \leq 2^{-k}.$$

We introduce a matrix variant of *Leftover Hash Lemma (LHL)* described in [GKPV10].

Lemma 2.1 (A Matrix Variant of LHL)**.** *Let $n = n(\lambda)$, $q = q(\lambda)$, and $\mathcal{D}$ be the distribution over $\mathbb{Z}_q^n$ with min-entropy k. For $m \log q \leq k - 2\lambda + \mathrm{O}(1)$,*

$$(\mathbf{Cs}, \mathbf{s}) \overset{s}{\approx} (\mathbf{u}, \mathbf{U}),$$

where $\mathbf{C} \xleftarrow{U} \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \xleftarrow{R} \mathcal{D}$, $\mathbf{u} \xleftarrow{U} \mathbb{Z}_q^m$, and $\mathbf{U} \xleftarrow{U} \mathbb{Z}_q^{m\times n}$.

We also recall the definition of *strong randomness extractors* [NZ96]. We borrow the definition from [DORS08].

Definition 2.5 (Strong Randomness Extractor)**.** *Let $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ be a probabilistic polynomial time function that uses r bits of randomness. We say that Ext is a $(m, 2^{-\lambda})$-strong randomness extractor if for all min-entropy m distributions $\mathcal{W}$ on $\{0,1\}^n$, $w \xleftarrow{R} \mathcal{W}$, $x \xleftarrow{U} \{0,1\}^r$, and $u \xleftarrow{U} \{0,1\}^l$*

$$(\mathsf{Ext}_x(w), x) \overset{s}{\approx} (u, x).$$

The extractor Ext can extract at most $l = m - 2\lambda + O(1)$ bits.

3 Underlying FHE

We now introduce the underlying encryption scheme. Our encoding system is constructed from FHE proposed by Gentry, Sahai and Waters [GGH13a]. We first introduce some algorithms to manipulate vectors, and then describe the entire scheme.

3.1 Vector Decomposition

The GSW-FHE scheme uses some algorithms to transform a vector. Let $\mathbf{g}^T := (1, 2, 2^2, \ldots, 2^{\lceil \log q \rceil - 1})$ and let $\mathbf{G} := \mathbf{g}^T \otimes \mathbf{I}_n$.

- $\mathsf{BitDecomp}_q(\mathbf{x})$: For a vector $\mathbf{x}^T = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, let $b_{i,j} \in \{0,1\}$ be such that $x_i = \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j \cdot b_{i,j}$. Output the vector

$$(b_{1,0}, \ldots, b_{1,\lceil \log q \rceil - 1}, \ldots, b_{n,0}, \ldots, b_{n,\lceil \log q \rceil - 1}) \in \{0,1\}^{n \cdot \lceil \log q \rceil}.$$

- $\mathsf{Powersof2}_q(\mathbf{y})$: For a vector $\mathbf{y} \in \mathbb{Z}_q^n$, output $\mathbf{y}^T\mathbf{G} \in \mathbb{Z}_q^{n\cdot\lceil\log q\rceil}$.
- $\mathsf{Combine}_q(\mathbf{z})$: For a vector $\mathbf{z} \in \mathbb{Z}_q^{n\cdot\lceil\log q\rceil}$, output $\mathbf{z}^T\mathbf{G}^T \in \mathbb{Z}_q^n$.
- $\mathsf{Flatten}_q(\mathbf{z})$: For a row vector $\mathbf{z} \in \mathbb{Z}_q^{n\cdot\lceil\log q\rceil}$, output $\mathsf{BitDecomp}_q(\mathsf{Combine}_q(\mathbf{z})) \in \{0,1\}^{n\cdot\lceil\log q\rceil}$.

For a matrix $\mathbf{X}$, let $\mathsf{BitDecomp}_q(\mathbf{X})$, $\mathsf{Combine}_q(\mathbf{X})$, or $\mathsf{Flatten}_q(\mathbf{X})$ be the matrix formed by applying the operation each row of $\mathbf{X}$ separately.

The above vector transformation algorithms satisfy the following conditions.

- $\langle \mathsf{BitDecomp}_q(\mathbf{x}), \mathsf{Powersof2}_q(\mathbf{y})\rangle = \langle \mathbf{x}, \mathbf{y}\rangle$.
- For any vector $\mathbf{z} \in \mathbb{Z}^N$,

$$\begin{aligned}\langle \mathbf{z}, \mathsf{Powersof2}_q(\mathbf{y})\rangle &= \langle \mathsf{Combine}_q(\mathbf{z}), \mathbf{y}\rangle \\ &= \langle \mathsf{Flatten}_q(\mathbf{z}), \mathsf{Powersof2}_q(\mathbf{y})\rangle.\end{aligned}$$

3.2 The GSW-FHE

The GSW-FHE scheme [GSW13] consists of six algorithms $\mathsf{GSW}.\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{MPDec}, \mathsf{Add}, \mathsf{Mult}\}$, which are defined as follows:

- $\mathsf{GSW.KeyGen}(1^\lambda, 1^\kappa)$: Choose a modulus $q := q(\lambda,\kappa)$, a lattice dimension $n := n(\lambda,\kappa)$, a parameter $m := O(n\log q)$, and a discrete Gaussian distribution $\chi := \chi(\lambda,\kappa)$ appropriately for the LWE assumption. Let $l := \lceil\log q\rceil$ and $N := (n+1)\cdot l$. Generate a matrix $\mathbf{A} \xleftarrow{\mathrm{U}} \mathbb{Z}_q^{m\times n}$ and a secret vector $\mathbf{s} \xleftarrow{\mathrm{U}} \mathbb{Z}_q^n$, and sample a noise vector $\mathbf{e} \xleftarrow{\mathrm{R}} \chi^m$. Compute $\mathbf{b} := \mathbf{As} + \mathbf{e} \mod q$, and set
$$\mathbf{B} := [\mathbf{b} \parallel \mathbf{A}] \in \mathbb{Z}_q^{m\times(n+1)}.$$
let $\mathbf{t} := \mathsf{Powersof2}_q(1, -\mathbf{s})$. Return $\mathsf{params}_{FHE} := (n, q, m, l, n, \chi)$, $\mathsf{pk} := \mathsf{BitDecomp}_q(\mathbf{B})$ and $\mathsf{sk} := \mathbf{t}$.
- $\mathsf{GSW.Enc}_{\mathsf{pk}}(\alpha)$: To encrypt a message α, choose a uniform matrix $\mathbf{R} \xleftarrow{\mathrm{U}} \{0,1\}^{N\times m}$ and return
$$\mathbf{C} := \mathsf{Flatten}(\alpha\cdot\mathbf{I}_N + \mathbf{R}\cdot\mathsf{BitDecomp}_q(\mathbf{B})) \in \{0,1\}^{N\times N}.$$
- $\mathsf{GSW.Dec}_{\mathsf{sk}}(\mathbf{C})$: We have
$$\mathbf{Ct} := \alpha\cdot\mathbf{t} + \hat{\mathbf{e}},$$
where $\hat{\mathbf{e}} = \mathbf{Re}$ is a small noise. Let $\mathbf{c}$ be the $l-1$th row of $\mathbf{C}$. Return 0 if $|\langle \mathbf{c}, \mathbf{t}\rangle| < q/8$, and 1 otherwise.
- $\mathsf{GSW.MPDec}_{\mathsf{sk}}(\mathbf{C})$: The first $l-1$ coefficients of $\mathbf{t}$ are $1, \ldots, 2^{l-2}$. Suppose that $\hat{\mathbf{e}}^t = (\hat{e}_1, \ldots, \hat{e}_{l-1})$ is a small noise vector, and $\mathbf{g}^t = (1, 2, \ldots, 2^{l-2})$, then the first $l-1$ coefficients of $\mathbf{Ct}$ are $\alpha\cdot\mathbf{g} + \hat{\mathbf{e}}$. Then we can extract the least significant bit of α from
$$\alpha\cdot 2^{l-2} + \hat{e}_{l-1} = \alpha_0\cdot 2^{l-2} + \hat{e}_{l-1} \mod 2^{l-1},$$
and the next least significant bit α_1 from $(\alpha - \alpha_0)\cdot 2^{l-3} + \hat{e}_{l-2} \mod 2^{l-1}$ as well.
- $\mathsf{GSW.Add}(\mathbf{C}_1, \mathbf{C}_2)$: To add two ciphertexts, output
$$\mathbf{C}_{add} := \mathsf{Flatten}(\mathbf{C}_1 + \mathbf{C}_2).$$
- $\mathsf{GSW.Mult}(\mathbf{C}_1, \mathbf{C}_2)$: To multiply two ciphertexts, return
$$\mathbf{C}_{mult} := \mathsf{Flatten}(\mathbf{C}_1 \cdot \mathbf{C}_2).$$

Definition 3.1 (Noise of the Ciphertext)**.** *For every $\mathbf{C} \in \{0,1\}^{N\times N}$, $\mathbf{s} \in \mathbb{Z}_q^n$, and $m \in \mathbb{Z}_q$, we define the noise of $\mathbf{C}$ as*
$$\mathsf{noise}_{\alpha,\mathbf{t}}(\mathbf{C}) := \| (\mathbf{C} - \alpha\cdot\mathbf{I}_N)\mathbf{t} \| .$$

4 Graded Encoding System on LWE

In this section, we first show the permuted key generation procedure for GSW-FHE, and then construct the graded encoding system on the LWE assumption.

4.1 Permuted Key Generation for GSW-FHE

Definition 4.1 (l-blockwise permutation)**.** *A permutation $\mathbf{P} \in \{0,1\}^{n\cdot l}$ is l-blockwise if for a $n\times n$ permutation matrix $\mathbf{P}_n \in \{0,1\}^{n\times n}$,*
$$\mathbf{P} = \mathbf{I}_l \otimes \mathbf{P}_n.$$

When we let $l := \lceil\log q\rceil$, we note that for any l-blockwise permutation $\mathbf{P}$ and matrix $\mathbf{X} \in \mathbb{Z}_q^{N\times N}$,
$$\mathsf{Flatten}_q(\mathbf{XP}) = \mathsf{Flatten}_q(\mathbf{X})\mathbf{P}.$$

The following permuted key generation procedure outputs a pair of keys jammed by the l-blockwise permutation.

- $\mathsf{GSW.KeyGen}'(1^\lambda, 1^\kappa)$: Choose a modulus $q := q(\lambda,\kappa)$, a lattice dimension $n := n(\lambda,\kappa)$, a parameter $m := O(n\log q)$, and a discrete Gaussian distribution $\chi := \chi(\lambda,\kappa)$ appropriately for the LWE assumption. Let $l := \lceil\log q\rceil$ and $N := (n+1)\cdot l$. Generate a matrix $\mathbf{A} \xleftarrow{\mathrm{U}} \mathbb{Z}_q^{m\times n}$ and a secret vector $\mathbf{s} \xleftarrow{\mathrm{U}} \mathbb{Z}_q^n$, and sample a noise vector $\mathbf{e} \xleftarrow{\mathrm{R}} \chi^m$. Compute $\mathbf{b} := \mathbf{As} + \mathbf{e} \mod q$, and set
$$\mathbf{B} := [\mathbf{b} \parallel \mathbf{A}] \in \mathbb{Z}_q^{m\times(n+1)}.$$
Let $\mathbf{t} := \mathsf{Powersof2}_q(1, -\mathbf{s}) \in \mathbb{Z}_q^N$. Choose a random l-blockwise permutation $\mathbf{P} \in \{0,1\}^{N\times N}$. Set $\mathbf{B}_{perm} = \mathsf{BitDecomp}_q(\mathbf{B})\cdot\mathbf{P}^T$ and $\mathbf{t}_{perm} = \mathbf{Pt}$. Return $\mathsf{params}_{FHE} := (n, q, m, l, n, \chi)$, $\mathsf{pk} := \mathbf{B}_{perm}$ and $\mathsf{sk} := \mathbf{t}_{perm}$.

Lemma 4.1. *For $(\mathbf{B}_{perm}, \mathbf{t}_{perm}) \xleftarrow{\mathrm{R}} \mathsf{GSW.KeyGen}'(1^\lambda, 1^\kappa)$ and $\alpha \in \mathbb{Z}_q$, let $\mathbf{C} = \mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(\alpha)$. We have*
$$\mathbf{Ct}_{perm} = \alpha\cdot\mathbf{t}_{perm} + \hat{\mathbf{e}},$$
where $\hat{\mathbf{e}}$ is a small noise vector.

Proof. For a random matrix $\mathbf{R} \xleftarrow{\mathrm{U}} \{0,1\}^{N\times m}$, we have $\mathbf{C} = \mathsf{Flatten}_q(\alpha\cdot\mathbf{I}_N + \mathbf{RB}_{perm})$.

$$\begin{aligned}\mathbf{Ct}_{perm} &= \mathsf{Flatten}_q(\alpha\cdot\mathbf{I}_N + \mathbf{RB}_{perm})\cdot\mathbf{Pt} \\ &= \mathsf{Flatten}_q(\alpha\cdot\mathbf{P} + \mathbf{R}\cdot\mathsf{BitDecomp}_q(\mathbf{B})\cdot\mathbf{P}^T\mathbf{P})\cdot\mathbf{t} \\ &= \alpha\cdot\mathbf{Pt} + \mathbf{R}\cdot(\mathsf{BitDecomp}_q(\mathbf{B})\mathbf{t}) \\ &= \alpha\cdot\mathbf{t}_{perm} + \hat{\mathbf{e}},\end{aligned}$$

where $\hat{\mathbf{e}} = \mathbf{Re}$. □

Lemma 4.2. *Let N be the integer that defines the form of ciphertexts (which are N by N matrices), T be the upper bound on fresh plaintexts, B be the upper bound on fresh ciphertexts, and $(\mathbf{B}_{perm}, \mathbf{t}_{perm})$ be a pair of keys generated by* $\mathsf{GSW.KeyGen}'$. *For $\alpha_1, \alpha_2 \in \mathbb{Z}_q$, let $\mathbf{C}_1 \xleftarrow{R} \mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(\alpha_1)$ and $\mathbf{C}_2 \xleftarrow{R} \mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(\alpha_2)$. Then we have*

$$\mathsf{GSW.Add}(\mathbf{C}_1, \mathbf{C}_2)\mathbf{t}_{perm} = (\alpha_1 + \alpha_2) \cdot \mathbf{t}_{perm} + \mathbf{e}_{add},$$
$$\mathsf{GSW.Mult}(\mathbf{C}_1, \mathbf{C}_2)\mathbf{t}_{perm} = (\alpha_1\alpha_2) \cdot \mathbf{t}_{perm} + \mathbf{e}_{mult},$$

where $\mathbf{e}_{add}$ and $\mathbf{e}_{mult}$ are the vectors with $\| \mathbf{e}_{add} \| < 2B$ and $\| \mathbf{e}_{mult} \| < (N+T)B$.

Proof. This is immediately from Lemma 4.1. □

4.2 Our Construction

Our graded encoding system consists of seven algorithms $\mathsf{LWEMMP.\{InstGen, Samp, Encode, Add, Mult, isZero, Extract\}}$.

Instance generation:
$(\mathsf{params}_{MMP}, \mathbf{p}_{zt}) \xleftarrow{R} \mathsf{LWEMMP.InstGen}(1^\lambda, 1^\kappa)$**.** The instance generation procedure first generates GSW-FHE parameters and keys

$$(\mathsf{params}_{FHE}, \mathbf{B}_{perm}, \mathbf{t}_{perm}) \xleftarrow{R} \mathsf{GSW.KeyGen}'(1^\lambda, 1^\kappa).$$

A secret integer z is chosen uniformly at random from $\mathbb{Z}_q$. Let $r := \log q + 3\lambda$. To encode at higher levels, we set $\mathbf{B}'_{perm} := \mathsf{BitDecomp}_q([\mathbf{As} \parallel \mathbf{A}])\mathbf{P}^T$ ($\mathbf{A}$ and $\mathbf{s}$ are used in $\mathsf{GSW.KeyGen}'$), choose $\mathbf{R}_i \xleftarrow{U} \{0,1\}^{N \times m}$ for $i = 1, \ldots, r$, and publish:

- $\{\mathbf{X}_i := \mathsf{Flatten}_q(\mathbf{R}_i\mathbf{B}'_{perm}/z)\}_{i \in [r]}$,
- $\mathbf{Y} := \mathsf{Flatten}_q(\mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(1)/z)$.

We choose a random seed s for a strong randomness extractor. The instance generation procedure $\mathsf{LWEMMP.InstGen}$ outputs parameters and a zero-testing parameter as:

- $\mathsf{params} := (n, q, m, N, \mathbf{B}_{perm}, \{\mathbf{X}_i\}_{i \in [r]}, \mathbf{Y}, s)$,
- $\mathbf{p}_{zt} := z^\kappa \cdot \mathbf{t}_{perm}$.

Sampling level-zero encodings:
$\mathbf{C} \xleftarrow{R} \mathsf{LWEMMP.Samp}(\mathsf{params}, \alpha)$**.** A level-0 encoding is a GSW-FHE ciphertext. To sample a level-0 encoding, this procedure takes as input a plaintext element $\alpha \in \mathbb{Z}_q$, and just outputs

$$\mathbf{C} \xleftarrow{R} \mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(\alpha).$$

Encoding at higher levels:
$\mathbf{U}_i \xleftarrow{R} \mathsf{LWEMMP.Encode}(\mathsf{params}, i, \mathbf{C})$**.** To encode a level-0 encoding (i.e., GSW-FHE ciphertext) at higher levels, we publish as part of our instance generation a level-1 random encoding of 1, $\mathbf{Y} = \mathbf{C}^{(1)}/z = \mathsf{GSW.Enc}_{\mathbf{B}_{perm}}(1)/z$. Given a level-0 encoding $\mathbf{C}_0$, we multiply it by $\mathbf{Y}$. That is, we compute $\mathbf{U}'_1 = \mathsf{GSW.Mult}(\mathbf{C}_0, \mathbf{Y})$. Note that we have $\mathbf{U}'_1 = \mathsf{Flatten}_q(\mathbf{C}_0\mathbf{Y})$.

In the GDDH assumption, every user keeps level-0 encodings secret and publishes level-1 encodings of the same underlying plaintext. In the above encoding procedure, however, it is insufficient to hide the level-0 encodings from the level-1 encodings: if $\det(\mathbf{Y}) \neq 0$, there exists the inverse matrix of $\mathbf{Y}$, therefore $\mathbf{C}_0$ can be recovered from $\mathbf{U}'_1 = \mathsf{Flatten}_q(\mathbf{C}_0\mathbf{Y})$. Instead we choose a random vector $\mathbf{f} := (f_1, \ldots, f_l) \in \mathbb{Z}_q^l$, and randomize $\mathbf{U}'_1$ with a linear combination of $\{\mathbf{X}_i\}_{i \in [r]}$:

$$\mathbf{U}_1 = \mathsf{Flatten}_q(\mathbf{U}'_1 + \sum_{i=1}^{r} f_i \cdot \mathbf{X}_i).$$

More generally, to generate a level-i encoding, we multiply $\mathbf{U}_1$ by $\mathbf{Y}$ $i-1$ times.

In the following two lemmas, we prove that the randomization term $\mathsf{Flatten}_q(\sum_{i=1}^{l} f_i \cdot \mathbf{X}_i)$ is statistically close to uniform over $\{0,1\}^{N \times N}$.

Lemma 4.3. *Let $\mathbf{P} \in \{0,1\}^{N \times N}$ be a l-blockwise permutation matrix, $\mathbf{A} \xleftarrow{U} \mathbb{Z}_q^{m \times n}$, and $\mathbf{s} \xleftarrow{U} \mathbb{Z}_q^n$. Set $\mathbf{B}' := [\mathbf{As} \parallel \mathbf{A}] \in \mathbb{Z}_q^{m \times (n+1)}$ and $\mathbf{B}'_{perm} := \mathsf{BitDecomp}_q(\mathbf{B})\mathbf{P}^T$, and choose $\mathbf{R} \xleftarrow{U} \{0,1\}^{N \times m}$. Then we have*

$$\mathsf{Flatten}_q(\mathbf{RB}'_{perm}) \overset{s}{\approx} \mathbf{U},$$

where $\mathbf{U} \xleftarrow{U} \{0,1\}^{N \times (n+1)}$.

Proof. Let $\mathbf{g}^T := (1, 2, 2^2, \ldots, 2^{l-1}) \in \mathbb{Z}^l$ and $\mathbf{G} := \mathbf{g}^T \otimes \mathbf{I}_n \in \mathbb{Z}^{n \times N}$. It holds that

$$\begin{aligned}\mathsf{Flatten}_q(\mathbf{RB}'_{perm}) &= \mathsf{BitDecomp}_q(\mathsf{Combine}_q(\mathbf{R}\mathsf{BitDecomp}_q(\mathbf{B}')))\mathbf{P}^T \\ &= \mathsf{BitDecomp}_q(\mathbf{R}\mathsf{BitDecomp}_q(\mathbf{B}')\mathbf{G}^T)\mathbf{P}^T \\ &= \mathsf{BitDecomp}_q(\mathbf{RB}')\mathbf{P}^T\end{aligned}$$

Since the multiplication of $\mathbf{P}^T$ does not change uniformity, $\mathbf{RB}' \overset{s}{\approx} \mathbf{U}'$ for $\mathbf{U}' \xleftarrow{U} \mathbb{Z}_q^{N \times (n+1)}$ implies that $\mathsf{Flatten}_q(\mathbf{RB}'_{perm}) \overset{s}{\approx} \mathbf{U}$. Let $\mathbf{r}_i^T \in \{0,1\}^m$ be the ith row vector of $\mathbf{R}$. For any $i \in [N]$, we prove that $(\mathbf{r}_i^T\mathbf{As}, \mathbf{r}_i^T\mathbf{A}) \overset{s}{\approx} (v_i, \mathbf{v}_i)$ for $v_i \xleftarrow{U} \mathbb{Z}_q$ and $\mathbf{v}_i \xleftarrow{U} \mathbb{Z}_q^n$. To prove this, we prove two statistical indistinguishabilities:

- $(\mathbf{r}_i^T\mathbf{As}, \mathbf{r}_i^T\mathbf{A}) \overset{s}{\approx} (\mathbf{v}_i^T\mathbf{s}, \mathbf{v}_i^T)$,
- $(\mathbf{v}_i^T\mathbf{s}, \mathbf{v}_i^T) \overset{s}{\approx} (v_i, \mathbf{v}_i)$.

They hold from the straightforward application of the leftover hash lemma (Lemma 2.1). Therefore $\mathbf{RB}'$ is statistically close to uniform, and so is $\mathsf{Flatten}_q(\mathbf{RB}'_{perm})$. □

Lemma 4.4. *Let λ be the security parameter and $r := \log q + 3\lambda$. For $i = 1, \ldots, r$, choose $\mathbf{R}_i \xleftarrow{U} \{0,1\}^{N \times m}$ and set $\{\mathbf{X}_i := \mathsf{Flatten}_q(\mathbf{R}_i\mathbf{B}'_{perm}/z)\}$. Choose $\mathbf{f}^T := (f_1, f_2, \ldots, f_r) \xleftarrow{U} \mathbb{Z}_q^r$. Then for $\mathbf{U} \xleftarrow{U} \mathbb{Z}_q^{N \times N}$ it holds that*

$$\mathsf{Flatten}_q(\sum_{i=1}^{r} f_i \cdot \mathbf{X}_i) \overset{s}{\approx} \mathbf{U}$$

Proof. Let $\mathbf{X}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,N})^T$ and $\mathbf{x}_{i,j}^T = (x_{i,j,1}, \ldots, x_{i,j,N})$. We have

$$\sum_{i=1}^{r} f_i \cdot \mathbf{X}_i = ((\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{r,1})\mathbf{f}, \cdots, (\mathbf{x}_{1,N}, \ldots, \mathbf{x}_{l,N})\mathbf{f}).$$

For any $i \in [r]$ and $j \in [N]$, $\langle \mathbf{f}, (x_{1,i,j}, \ldots, x_{r,i,j}) \rangle$ is statistically close to uniform over $\mathbb{Z}_q$ from the parameter settings and the leftover hash lemma. □

Adding and Multiplying encodings:
$\mathsf{LWEMMP.}\{\mathsf{Add}(\mathsf{params}, \mathbf{u}_i, \mathbf{u}_i'), \mathsf{Mult}(\mathsf{params}, \mathbf{u}_i, \mathbf{u}_i')\}$**.** The level-$i$ encodings are GSW-FHE ciphertexts divided by some integer z^i. To output the addition/multiplication of the input encodings, the procedures just respectively run GSW.Add and GSW.Mult that take the two encodings as inputs.

Zero testing:
$\mathsf{LWEMMP.isZero}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}_\kappa) \stackrel{?}{=} 0/1$**.** Because of the additive homomorphism of encodings, we can test equality between encodings by subtracting them and testing for zero. To test if $\mathbf{U}_\kappa = \mathbf{C}/z^\kappa$ encodes 0, we just multiply it by the zero-testing parameter $\mathbf{p}_{zt}$, and check whether the resulting vector $\mathbf{w} := \mathbf{U}_\kappa \mathbf{p}_{zt}$ is short. That is,

$$\mathsf{LWEMMP.isZero}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}_\kappa) = \begin{cases} 1 & \text{if } \| \mathbf{w} \| < \frac{q}{8} \\ 0 & \text{otherwise.} \end{cases}$$

For $\mathbf{R} \in \{0,1\}^{N \times m}$ and $\mathbf{U}_\kappa = \mathbf{C}/z^\kappa = \mathsf{Flatten}_q(\alpha \cdot \mathbf{I}_n + \mathbf{R}\mathbf{B}_{perm})/z^\kappa$, we have

$$\mathbf{w} = \mathbf{U}_\kappa \mathbf{p}_{zt} = \frac{\mathbf{C}}{z^\kappa} \cdot z^\kappa \cdot \mathbf{t}_{perm} = \mathbf{C}\mathbf{t}_{perm} = \alpha \cdot \mathbf{t}_{perm} + \mathbf{e}.$$

Therefore $\mathbf{w} = \mathbf{e}$ is small if $\alpha = 0$, and $\mathbf{w}$ must be large otherwise.

Lemma 4.5 (Correctness of Zero-testing)**.** *For the modulus q chosen as in the instance generation, let $\mathbf{U}_\kappa^{(\alpha)}$ be a level-κ encoding of $\alpha \in \mathbb{Z}_q$, and $\mathbf{p}_{zt}$ be the zero-testing parameter. We have $\| \mathbf{U}_\kappa^{(0)} \mathbf{p}_{zt} \| < q/8$.*

Proof. This is immediately from the above argument and the parameter settings. □

Extraction: $s \stackrel{R}{\leftarrow} \mathsf{LWEMMP.Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}_\kappa)$. To extract the random function of the plaintext element in a level-κ encoding $\mathbf{U}_\kappa = \mathbf{C}_\kappa/z^\kappa$, we just multiply it by $\mathbf{p}_{zt}$, collect the three most significant bits of the N coefficients of the result, and apply a strong randomness extractor for them. That is,

$$\mathsf{LWEMMP.Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U}_\kappa) = \mathsf{Ext}_s(\mathsf{msbs}_3(\mathbf{U}_\kappa \mathbf{p}_{zt})),$$

where Ext_s is a $(\lfloor \log q \rfloor, 2^{-\lambda})$-strong randomness extractor, msbs_3 extracts the three most significant bits of the result.

Lemma 4.6. *Let $\mathbf{p}_{zt}$ be a zero-testing parameter, and let $\mathsf{Ext} : \{0,1\}^{3N} \to \{0,1\}^{\lfloor \log q \rfloor - 2\lambda + O(1)}$ be a $(\lfloor \log q \rfloor, 2^{-\lambda})$-strong randomness extractor. For any $\alpha, \alpha' \in \mathbb{Z}_q$, let $\mathbf{U}_\kappa^{(\alpha)}$ and $\mathbf{U}_\kappa^{(\alpha')}$ be their encodings at the level κ, respectively. If $\alpha = \alpha'$, then $\mathsf{Ext}_s(\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha)}))$ and $\mathsf{Ext}_s(\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha')}))$ extract the same nearly uniform bit-string.*

Proof. If $\alpha = \alpha'$, by Lemma 4.5, we have

$$\| \mathbf{U}_\kappa^{(\alpha)} \mathbf{p}_{zt} - \mathbf{U}_\kappa^{(\alpha')} \mathbf{p}_{zt} \| = \| (\mathbf{U}_\kappa^{(\alpha)} - \mathbf{U}_\kappa^{(\alpha')}) \mathbf{p}_{zt} \| < \frac{q}{8},$$

and so expect $\mathbf{U}_\kappa^{(\alpha)} \mathbf{p}_{zt}$ and $\mathbf{U}_\kappa^{(\alpha')} \mathbf{p}_{zt}$ to agree on their $\log q - \log q/8 = 3$ most significant bits [1]. Let $\mathcal{X}$ be the probability distribution of $\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha)} \mathbf{p}_{zt})$ (and $\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha')} \mathbf{p}_{zt})$). Then we have $H_\infty(\mathcal{X}) \geq \lfloor \log q \rfloor$. Therefore $\mathsf{Ext}_s(\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha)} \mathbf{p}_{zt}))$ and $\mathsf{Ext}_s(\mathsf{msbs}_3(\mathbf{U}_\kappa^{(\alpha')} \mathbf{p}_{zt}))$ can extract the same nearly uniform bit-string of length at most $\lfloor \log q \rfloor - 2\lambda + O(1)$. □

4.3 Parameter Settings

We list constraints of the parameters for our maps.

- Consider the multipartite Diffie-Hellman key exchange (mDH-KE) (See Appendix A for details.) setting. Let $\mathbf{C}$ be a level-0 encoding with plaintext α, T be the upper bound on the fresh plaintexts, and B be the upper bound on the noise of fresh ciphertexts (i.e., level-0 encodings). Suppose $\mathbf{U}_1 = \mathsf{Flatten}(\mathbf{C}_1/z) \stackrel{R}{\leftarrow} \mathsf{LWEMMP.Enc}(\mathsf{params}, \mathbf{C}, 1)$, then

  $$\mathsf{noise}_{\alpha,\mathbf{s}}(\mathbf{C}_1) \leq (N+1)B.$$

  As described in [BV13], multiplication of GSW-FHE ciphertexts increases the noise in an asymmetric manner. The product of κ level-1 encodings $\mathbf{U}_\kappa = \mathsf{Flatten}(\mathbf{C}_\kappa/\mathbf{z})$ (which has the plaintext α_κ) has the noise at most

  $$\mathsf{noise}_{\alpha_\kappa,\mathbf{s}}(\mathbf{C}_\kappa) \leq (T^{\kappa-1}N^2 + T^\kappa N + T^{\kappa-1})B.$$

  To keep the correctness of the multilinear maps operations (addition and multiplication) the right side of the inequality needs to be less than $q/8$.

- In our construction, when given an encoding, an adversary can obtain a set of elements that includes the plaintext of the encoding. In the mDH-KE with κ parties, to avoid the obvious attack, we should take $n = \Omega(2^{\lambda/\kappa})$.

- For the sake of the 2^λ security against known attacks, n (and so N) must increase linearly with $\log(q/B)$. This implies that q/B grows more like $\exp(\kappa + \log \kappa)$.

- Applying the leftover hash lemma in Lemma 4.4 requires that $r := \log q + 3\lambda$.

- The security of GSW-FHE requires that the rank m of the public matrix $\mathbf{A}$ ought to be $2n \log q$.

4.4 A Note on Some Decisional Problems against the GGH's Weak Discrete Log Attack

The *weak discrete log* attack [GGH13a] is to compute a randomized plaintext element of a challenge at the strictly lower level than κ from public information (e.g., the challenge in the security assumption, the zero-testing parameter, and the level-1 encodings of 1 and 0).

[1] Otherwise, we must have $\| (\mathbf{U}_\kappa^{(\alpha)} - \mathbf{U}_\kappa^{(\alpha')}) \mathbf{p}_{zt} \| \geq q/8$, and hence the three most significant bits of the corresponding $\mathbf{U}_\kappa \mathbf{p}_{zt}$ and $\mathbf{U}_\kappa' \mathbf{p}_{zt}$ must be different.

In the above attack, as described in [CLT13], the GGH encodings [GGH13a] do not support some decisional problems (e.g., the decisional subgroup problem using composite-order maps, and the decisional linear (DLIN) problem) in the lower level (below κ), while the CLT construction does so.

Since the LWE assumption implies that the randomized plaintext element leaks no information about the plaintext in our construction, it seems to have the tolerance for the weak discrete log attack in the DLIN and subgroup problems.

4.5 Security

We instantiate the GDDH assumption for our maps, which can be described like below.

1. $(\mathsf{params} = (n, q, m, N, \mathbf{B}, \mathbf{X}, \mathbf{Y}), \mathbf{p}_{zt}) \xleftarrow{R} \mathsf{LWEMMP.InstGen}(1^\lambda, 1^\kappa)$.
2. **For** $i = 1, \ldots, \kappa + 1$,
   (i) $\alpha_i \xleftarrow{U} \mathbb{Z}_q$.
   (ii) $\mathbf{A}_i \xleftarrow{R} \mathsf{GSW.Enc}_{\mathbf{B}}(\alpha_i)$.
   (iii) $\mathbf{R} \xleftarrow{U} \{0,1\}^{N \times m}$.
   (iv) $\mathbf{U}_i = \mathbf{A}_i \mathbf{Y} + \mathsf{BitDecomp}_q(\mathbf{RX})$.
3. $\alpha \xleftarrow{U} \mathbb{Z}_q$.
4. $\hat{\mathbf{A}} \xleftarrow{R} \mathsf{GSW.Enc}_{\mathbf{B}}(\alpha)$.
5. $\tilde{\mathbf{U}} = \mathbf{A}_{\kappa+1} \times \prod_{i=1}^{\kappa} \mathbf{U}_i$.
6. $\hat{\mathbf{U}} = \hat{\mathbf{A}} \times \prod_{i=1}^{\kappa} \mathbf{U}_i$.

Definition 4.2 (GDDH)**.** *The GDDH problem is to distinguish the following two distributions:*

$$\mathcal{D}_{GDDH} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{\mathbf{U}_i\}_{i \in [\kappa+1]}, \tilde{\mathbf{U}})\}$$
$$\mathcal{D}_{RAND} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{\mathbf{U}_i\}_{i \in [\kappa+1]}, \hat{\mathbf{U}})\}.$$

The GDDH assumption states that for any efficient adversary $\mathcal{A}$ and a security parameter λ,

$$|Pr[\mathcal{A}(\mathcal{D}_{GDDH}) \to 1] - Pr[\mathcal{A}(\mathcal{D}_{RAND}) \to 1]| < \mathsf{negl}(\lambda).$$

5 Conclusion

We constructed the multilinear maps on LWE. The security of our maps was based on the DDH-like assumption, which is called the graded decisional Diffie-Hellman (GDDH) assumption.

An interesting direction for future work is to reduce the GDDH assumption for our maps to established assumptions, such as LWE.

References

[BGK+13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631, 2013. http://eprint.iacr.org/2013/631.

[BL13] Alexandra Berkoff and Feng-Hao Liu. Leakage resilient fully homomorphic encryption. Cryptology ePrint Archive, Report 2013/822, 2013. http://eprint.iacr.org/2013/822.

[BR13a] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. *Advances in Cryptology - CRYPTO 2013, LNCS*, 8043:416–434, 2013.

[BR13b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563, 2013. http://eprint.iacr.org/2013/563.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[BV13] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. Cryptology ePrint Archive, Report 2013/541, 2013. http://eprint.iacr.org/2013/541.

[CCK+13] Jung Hee Cheon, Jean-Sébastian Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. *Advances in Cryptology - EUROCRYPT 2013, LNCS*, 7881:315–335, 2013.

[CLT13] Jean-Sébastian Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. *Advances in Cryptology - CRYPTO 2013, LNCS*, 8042:476–493, 2013.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[Gen09a] Craig Gentry. *A FULLY HOMOMORPHIC ENCRYPTION SCHEME*. PhD thesis, Stanford University, 2009. http://crypto.stanford.edu/craig.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. *STOC*, pages 169–178, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. *Advances in Cryptology - EUROCRYPT 2013, LNCS*, 7881:1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *FOCS*, 2013.

[GKPV10] Shafi Goldwasser, Yael Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. *ICS*, 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. How to use a short basis: Trapdoors for hard lattices and new cryptographic constructions. *STOC*, pages 197–206, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Advances in Cryptology - CRYPTO 2013, LNCS*, 8042:75–92, 2013.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *Advances in Cryptology - EUROCRYPT 2012, LNCS*, 7237:700–718, 2012.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and Systeme Sciences*, 52(1):43–53, 1996.

[PTS13] Rafael Pass, Sidharth Telang, and Karn Seth. Obfuscation from semantically-secure multilinear encodings. Cryptology ePrint Archive, Report 2013/781, 2013. `http://eprint.iacr.org/2013/781`.

[Reg05] Oded Regev. On lattice, learning with errors, random linear codes, and cryptography. *STOC*, pages 84–93, 2005.

A Multipartite Diffie-Hellman Key Excahnge

In [BS03], Boneh and Silverberg displayed how to implement multipartite Diffie-Hellman key exchange from multilinear maps. As in [GGH13a, CLT13], our construction can be used to construct a one round N-way Diffie-Hellman key exchange.

Let us consider N parties willing to share a secret string s in the one-round [2] protocol. We recall the construction by [GGH13a].

$\mathsf{Setup}(1^\lambda, 1^N)$. Takes as input a security parameter λ and the number of party N, outputs $(\mathsf{params}, \mathbf{p}_{zt}) \xleftarrow{R} \mathsf{InstGen}(1^\lambda, 1^{N-1})$ as the public parameters for the level $N-1$.

$\mathsf{Publish}(\mathsf{params}, \mathbf{p}_{zt}, i)$. Each party P_i generates a random level-0 encoding $\mathbf{C}_i \xleftarrow{R} \mathsf{Samp}(\mathsf{params})$ as a secret key, and publishes as a public key the corresponding level-1 encoding $\mathbf{U}_i \xleftarrow{R} \mathsf{Encode}(\mathsf{params}, i, \mathbf{C}_i)$.

$\mathsf{KeyGen}(\mathsf{params}, \mathbf{p}_{zt}, j, \mathbf{d}_j, \{\mathbf{w}_i\}_{j\neq i})$. Each party P_i computes $\mathbf{U} = \mathbf{C}_i \times \prod_{j\neq i} \mathbf{U}_j$ by the multiplication procedure Mult. To obtain a common secret string s, P_i invokes the extraction routine, $s \xleftarrow{R} \mathsf{Extract}(\mathsf{params}, \mathbf{p}_{zt}, \mathbf{U})$.

Theorem A.1 (**Theorem 2**, [GGH13a])**.** *The protocol described above is a one-round N-way Diffie-Hellman key exchange protocol if the GDDH assumption holds for the underlying encoding scheme.*

[2] The word "one-round" refers to the setting in which each party can broadcast one value to all other parties.