Universitatea “Alexandru Ioan Cuza” din Iaşi
Facultatea de Informatică

LUCRARE DE LICENŢĂ

Multilinear Maps over Ideal Lattices

propusă de

Student: Ciprian Băetu
Coordonator ştiinţific: Prof. Dr. Ferucio Laurenţiu Ţiplea

Sesiunea: iunie - iulie
2017

Universitatea “Alexandru Ioan Cuza” din Iaşi
Facultatea de Informatică

Multilinear Maps over Ideal Lattices

Student: Ciprian Băetu
Coordonator ştiinţific: Prof. Dr. Ferucio Laurenţiu Ţiplea

Sesiunea: iunie - iulie
2017

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul "Multilinear Maps over Ideal Lattices" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi n traducere proprie din altă limb, sunt scrise ntre ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de ctre alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,
24 iunie 2017

Absolvent,
Băetu Ciprian

(semnătura în original)

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul "Multilinear Maps over Ideal Lattices", codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza din Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,
24 iunie 2017

Absolvent,
Băetu Ciprian

(semnătura în original)

Contents

Introduction **4**
Contribution . . . 5
Organization . . . 5

1 Multilinear Maps and Graded Encoding Systems **6**
1.1 Notations . . . 6
1.2 Bilinear Maps . . . 7
1.3 Cryptographic Multilinear Maps . . . 7
1.3.1 From Self-Bilinear to Multilinear Maps . . . 7
1.3.2 Efficient Procedures . . . 8
1.3.3 Hardness Assumptions . . . 9
1.4 Graded Encoding Systems . . . 9
1.4.1 Efficient Procedures . . . 10
1.4.2 Hardness Assumptions . . . 12

2 Mathematical Background **13**
2.1 Algebra . . . 13
2.2 Lattices . . . 14
2.2.1 General Influence . . . 14
2.2.2 Basic Concepts . . . 14
2.2.3 Hard problems . . . 16
2.2.4 Results concerning short vectors . . . 16
2.2.5 LLL Algorithm . . . 17
2.2.6 Ideal Lattices . . . 19
2.3 Probabilities and Statistics . . . 19

3 Proposed Encoding Scheme **21**
3.1 Overview . . . 21
3.2 Efficient Procedures . . . 21
3.3 Parameter constraints . . . 27
3.4 Hardness Assumptions . . . 28

4 Security **30**

Conclusions **31**

Introduction

Usually, cryptographic primitives are constructed under the assumption that several problems are intractable, i.e. there exist no polynomial-running time algorithm to solve them. Vercauteren [21] realized an extensive research regarding the intractable problems that are most used in cryptography, such as integer factoring, discrete logarithm, computational Diffie-Hellman, shortest vector problem and many others.

In the last ten years, bilinear maps proved to be very useful in cryptography. Making use of the interesting properties of such maps, cryptographers managed to construct schemes for one-round three-party key exchange [16], identity based encryption [4] and many other applications. After the moment that bilinear maps proved to be undoubtedly useful, researchers have tried to generalize the concept. Thus, multilinear maps were defined and the search for their applications has begun. Boneh and Silverberg [5] showed that symmetric multilinear maps can be used to realize a one-round multi-party key exchange scheme but, after their attempts to construct such maps failed, they drew the conclusion that "such maps might have to either come from outside the realm of algebraic geometry, or occur as 'unnatural' computable maps arising from geometry."

Garg, Gentry and Halevi [11] proposed a construction based on lattices that approximate the multilinear maps in hard-discrete-logarithm groups. Using this candidate, they could construct an application to multipartite Diffie-Hellman key exchange scheme and also the first construction of Attribute-Based Encryption for general circuits [12]. A short period after the aforementioned candidate was proposed, Coron, Lepoint and Tibouchi [9] created a similar construction, based on integers instead of lattices.

However, these construction proved to be susceptible to attacks, and a devastating zeroizing attack for the integer construction is presented thoroughly in [7]. Numerous fixing tentatives of these schemes were designed, but for each of them there was found at least another attack. Therefore currently, new methods of constructing multilinear maps and Graded Encoding Schemes still constitute an open field, very interesting for cryptographers.

Contribution

This work represents a survey over the bilinear and multilinear maps and their applications. Furthermore, in this paper is presented the concept of Graded Encoding Scheme, a modality of approximating multilinear maps. The core of the paper is represented by the review of the lattice-based construction, designed by Garg, Gentry and Halevi in [11].

Organization

The paper is divised into 5 sections. First one - introduction bla-bla. In the second section, bla bla bla ... etc etc.

Chapter 1

Multilinear Maps and Graded Encoding Systems

In this chapter, multilinear maps are defined and also, the particular case of bilinear maps is discussed, along with results concerning self-bilinear applications. Thereafter, *Graded Encoding Schemes* are defined, as an approximate to multilinear maps.

Observation: Regarding the multilinear applications and Graded Encoding Systems schemes, and also for the lattice-based candidate designed in [11], the paper encompasses one subsection of efficient procedures, and another one of hardness assumptions. The reader should be aware of this detail and realize the analogy and differences of the mentioned schemes.

1.1 Notations

Throughout the paper, the following notations will be used, as mentioned in [5]:

1. The set of all finite length binary string is denoted by $\{0,1\}^*$. Also, for any positive integer n, the notation $\{0,1\}^n$ denotes the set of all binary strings of length n.

2. Let $\mathcal{D}$ be a probabilistic event. Then, the **probability** of the event is denoted by $\Pr[\mathcal{D}]$.

3. Let m be an arbitrary positive integer. A **randomized algorithm** $\mathcal{A}$ is a function of two parameters, $x \in \{0,1\}^*$ - the input of the algorithm and $r \in \{0,1\}^n$ - a sequence of random bits used by the algorithm Also, for an input $x, \mathcal{A}(x)$ denotes the random variable $\mathcal{A}(x,R)$, where R is uniformly distributed in $\{0,1\}^m$.

4. Let S be a finite set. Then, by $x \leftarrow S$ is defined a random variable x that picks an element of S at random, i.e. $\Pr[x=s] = \frac{1}{|S|}, \forall s \in S$.

5. Let $\mathcal{A}$ be a randomized algorithm and let $x \in \{0,1\}^*$. Then, $y \leftarrow \mathcal{A}(x)$ denotes the random variable y that represents the output of the algorithm $\mathcal{A}$ on input x.

6. Let X and Y be two distributions over domain D. Then, the **statistical distance** between X and Y is:

$$\Delta(X,Y) = \frac{1}{2}\sum_{d\in D} |\Pr[X=d] - \Pr[Y=d]|.$$

7. A function $f : \mathbb{Z}^+ \to \mathbb{R}^+$ is said to be **negligible** if:

$$\forall c > 0, \exists n_c \in \mathbb{Z}^+ \text{ such that } f(n) < \frac{1}{n^d}, \forall n \geq n_c.$$

1.2 Bilinear Maps

As stated before, bilinear maps are a specific case of multilinear maps. They proved to be a highly useful tool in cryptography, with many applications, such as: tripartite protocol [16], identity based encryption [4] and Attribute-based encryption scheme for monotone boolean formulas [15]. In this section, bilinear maps are only defined, while next section presents a relationship between self-bilinear maps and multilinear maps.

Definition 1 (Bilinear Map [2]). *Given the cyclic groups G and G_t (written additively) of the same order p, a (symmetric) map $e : G \times G \to G_t$ is said to be bilinear if the following properties hold:*

1. ***(Bi-linearity)*** *$e({g_1}^{x_1}, {g_2}^{x_2}) = e(g_1, g_2)^{x_1 x_2}$, for any $x_1, x_2 \in \mathbb{Z}_p$ and any $g_1, g_2 \in G$;*
2. ***(Non-degeneracy)*** *If $g_1, g_2 \in G$ are generators of G, then $e(g_1, g_2)$ is a generator of G_t;*
3. ***(Efficient computability)*** *There exists a polynomially-bounded algorithm to compute $e(g_1, g_2)$, for any $g_1, g_2 \in G$.*

1.3 Cryptographic Multilinear Maps

Definition 2 (Multilinear Maps [20]). *Let $k \geq 2$ be an integer number and $G_1, G_2, ..., G_k, G_T$ be $k+1$ cyclic groups (written additively), of same order p. Then, a $k-$multilinear map is a mapping $e : G_1 \times ... \times G_k \to G_T$, with the following properties:*

1. ***(Linearity)*** *For every $g_1 \in G_1, ..., g_k \in G_k$, every $i \in \{1, 2, .., k\}$ and every $\alpha \in \mathbb{Z}_p$, it holds that:*

$$e(g_1, ..., \alpha \cdot g_i, .., g_k) = \alpha \cdot e(g_1, ..., g_k))$$

2. ***(Non-degeneracy)*** *If $g_1 \in G_1, ..., g_k \in G_k$ are generators of their respective groups, then $e(g_1, ..., g_k)$ is a generator of G_T.*

1.3.1 From Self-Bilinear to Multilinear Maps

Definition 3. *A self-bilinear map is a bilinear map where the domain and target groups are the same.*

Proposition 1. *Let G be a cyclic group of order p and $e : G \times G \to G$ be a self-bilinear map. Therefore, a $k-$multilinear map $e_k : G^k \to G$ can be constructed from e, for any $k \geq 2$.*

Proof. The proof is realized by induction. First, for the base case $k = 2$, it is trivial to observe that e itself is a 2-liniar map. Then, suppose that an $n-$multilinear

map $e_n : G^n \to G$ can be constructed starting from e, and it can be easily shown that a $(n+1)$-multilinear map $e_{n+1} : G^{n+1} \to G$ can be constructed, as follows:

$$e_{n+1}(g_1, .., g_n, g_{n+1}) = e(e_n(g_1, .., g_n), g_{n+1}), \forall g_1, .., g_{n+1} \in G.$$

Indeed, from the fact that e_n is multilinear it follows that, for any $g_1 \in G_1, ..., g_n \in G_n$, any $i \in \{1, .., n\}$ and any $\alpha \in \mathbb{Z}_p$, $e_n(g_1, ..., \alpha \cdot g_i, .., g_n) = \alpha \cdot e_n(g_1, ..., g_n))$. Using the bilinearity of e, it results that e_{n+1} respects the **linearity** condition.
Let $g_1, ..., g_n$ be generators of G. Then, using the fact that e_n is $n-$multilinear, it follows that $e_n(g_1, .., g_n)$ is also a generator of G. Corroborating the last result with the non-degeneracy property of e, it ensues that e_{n+1} respects **non-degeneracy** condition, from which the conclusion that e_{n+1} is a $(n+1)$-multilinear map can be drawn. □

However, Cheon and Lee [8] proved that self-bilinear maps on prime order groups do not exist, except that the computational Diffie-Hellman problem is easy. That is the main motivation of [2], which analyzes the existence of self-bilinear maps on groups of composite order.

1.3.2 Efficient Procedures

In order to use the cryptographic multilinear applications in a real-world environment, efficient procedures must be designed, to be evaluated by computers. Therefore, as specified in [11] a cryptographic multilinear map scheme is a 5-uple $\mathcal{MMP}$ = **(InstGen, EncTest, add, neg, map)** that is described below:

(a) **Instance Generation.** A procedure with a "factory" role must exist, in order to instantiate the parameters of the scheme. This procedure is **InstGen**.

- **Input:** λ - the security parameter and $k \geq 2$ - the multilinearity parameter.
- **Output:** (**params**, $g_1, .., g_k$), where **params** $= (G_1, .., G_T, p, e)$. Here $G_1, .., G_k$, G_T represent the groups, $p \in \mathbb{Z}$ is their order, e is the representation of the multilinear map and $g_i \in \{0,1\}^*$ is the representation of a generator of G_i, for every $i \in \{1, .., k\}$.

(b) **Element Encoding.** A procedure that decides if a sequence of bits represents an encoding of an element in one of the groups must be defined, and it is named **EncTest**.

- **Input: params** - the instance parameters, $i \in \{1, .., k+1\}$ - index of the desired group and $x \in \{0,1\}^*$ - encoding of the tested element.
- **Output:** True, if x is a valid encoding of an element in G_i, False otherwise. *Note:* The extension $G_{k+1} = G_T$ is performed.

(c) **Group addition.** The procedure **add** simply applies the group operation upon two provided elements representations.

- **Input: params** - the instance parameters, $i \in \{1, .., k+1\}$ - index of the desired group, x, y - representations of elements to be added.
- **Output:** the representation of $x + y \in G_i$.

(d) **Group negation.** The procedure **neg** returns the inverse representation of the element provided as parameter.

- **Input: params** - the instance parameters, $i \in \{1, .., k+1\}$ - index of the desired group, x - representation of the element to be negated.
- **Output:** the representation of $-x \in G_i$.

(e) **Map computation.** The procedure **map** returns the representation of the multilinear mapping over the elements given as parameters.

- **Input: params** - the instance parameters, $x_1 \in G_1, .., x_k \in G_k$ - elements in domain groups.
- **Output:** the representation of $e(x_1, .., x_k) \in G_T$.

1.3.3 Hardness Assumptions

For the multilinear map to be used in cryptography, it is inquired that at least the Multilinear Discrete Logarithm problem (MDL) and Multilinear Decisional Diffie-Hellman problem (MDDH) to be hard in the used groups. The specified problems are reminded below:

1. **Multilinear Discrete Logarithm (MDL [11]).** It is said that the MDL problem is hard for a multilinear map scheme $\mathcal{MMP}$ if, for any $k > 1$, any $i \in \{1, ..., k\}$ and all probabilistic polynomial running time algorithms, the discrete logarithm advantage of an adversary $\mathcal{A}$,

$$\mathrm{AdvDlog}_{\mathcal{MMP},\mathcal{A},k}(\lambda) \stackrel{\mathrm{def}}{=} Pr[\mathcal{A}(params, i, g_i, \alpha \cdot g_i) = \alpha : (params, g_1, ..., g_k) \leftarrow \mathrm{InstGen}(1^\lambda, 1^k), \alpha \leftarrow \mathbb{Z}_p],$$

   is negligible in λ.

2. **Multilinear DDH (MDDH [11]).** The MDDH problem is hard for a symmetric multilinear map scheme $\mathcal{MMP}$ (with $G_1 = .. = G_k = G_T$) if for any probabilistic polynomial running time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the distributions:

$$(params, g, \alpha_0 g, \alpha_1 g, .., \alpha_k g, (\prod_{i=0}^{k} \alpha_i) \cdot e(g, .., g)) \text{ and}$$
$$(params, g, \alpha_0 g, \alpha_1 g, .., \alpha_k g, \alpha \cdot e(g, .., g))$$

   is negligible in λ, where $(params, g) \leftarrow \mathrm{InstGen}(1^\lambda, 1^k)$ and $\alpha, \alpha_1, .., \alpha_k$ are uniformly random in $\mathbb{Z}_p$.

1.4 Graded Encoding Systems

Garg, Gentry and Halevi formally defined the Graded Encoding Systems in [11]. Using the mentioned system, the authors managed to realize an "approximation" of the sought after multilinear maps in groups in which the DL problem is hard.

They generalize the conventional constructions, by replacing the usual exponent space, $\mathbb{Z}_p$, with a generic algebraic ring or field R. Also, the system is non-deterministic, with the significance that the same element can be encoded in plentiful of ways. Another difference is that the system offers the possibility of "partial mapping", i.e. multiplying any

number of encodings, not only k, as in the multilinear map case. Thus, the structure of the system is much richer, revealing the opportunity to encode the same element on many different levels.

In the current section, the general settings of the system are discussed, following that the construction of an instance of Graded Encoding Systems (GES) to be approached in a subsequent chapter.

Definition 4 ($k-$ Graded Encoding System). *Let $k > 1$ be an integer. A $k-$ Graded Encoding System is formed by a ring $(R, +_R, \cdot_R)$ and a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : \alpha \in R, i \in \{0,1,..,k\}\}$, with the properties:*

1. *For any $i \in \{0,1,..,k\}$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint;*
2. *An associative binary operation '+' and an unary operation '-' can be defined on $\{0,1\}^*$, such that for any $\alpha_1, \alpha_2 \in R$, any $i \in \{0,..,k\}$ and any encodings $u_1 \in S_i^{(\alpha_1)}$, $u_2 \in S_i^{(\alpha_2)}$, it follows that $u_1 + u_2 \in S_i^{(\alpha_1 +_R \alpha_2)}$ and $-u_1 \in S_i^{(-_R \alpha_1)}$;*
3. *An associative binary operation '$\times$' can be defined on $\{0,1\}^*$, such that for any $\alpha_1, \alpha_2 \in R$, any integers $0 \leq i_1, i_2$ such that $i_1 + i_2 \leq k$ and any encodings $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$, it results that $u_1 \times u_2 \in S_{i_1 + _R i_2}^{(\alpha_1 \cdot_R \alpha_2)}$.*

1.4.1 Efficient Procedures

(a) **Instance Generation.** Again, **InstGen** is a randomized procedure that has the task to instantiate the parameters of the scheme.

- **Input:** $\boldsymbol{\lambda}$ - the security parameter and $\boldsymbol{k} \geq 2$ - the multilinearity parameter.
- **Output:** $\mathbf{InstGen}(1^\lambda, 1^k) = (\mathbf{params}, \mathbf{p}_{zt})$, where **params** completely specifies the $k-$GES, and $\mathbf{p}_{zt}$ is a zero-test parameter, as described below.

(b) **Ring Sampler. samp** is a non-deterministic procedure that returns a "level-zero" encoding of a nearly uniform element of R.

- **Input**: None
- **Output:** $\mathbf{samp}(\mathbf{params}) = \boldsymbol{a} \in S_0^{(\alpha)}$, with $\alpha \in R$ - nearly uniform.

(c) **Encoding.** The procedure **enc** computes an encoding on any level of a given "level-zero" encoding.

- **Input**: **params** - the instance parameters, $\boldsymbol{i} \in \{0,..,k\}$ - the index of the desired level of encoding and $\boldsymbol{a} \in S_0^{(\alpha)}$ - the "level-zero" encoding of an element $\alpha \in R$.
- **Output:** $\mathbf{enc}(\mathbf{params}, i, a) = \boldsymbol{v} \in S_i^{(\alpha)}$ - a level$-i$ encoding of the same α previously specified.

(d) **Addition.** The procedure **add** computes an encoding of the sum of two same-level encodings.

- **Input**: **params** - the instance parameters, $\boldsymbol{i} \in \{0,..,k\}$ - the index of the level of encoding, $\boldsymbol{v_1} \in S_i^{(\alpha_1)}$, $\boldsymbol{v_2} \in S_i^{(\alpha_2)}$ (where $\alpha_1, \alpha_2 \in R$) - encodings of the elements to be added.

- **Output: add**(**params**, i, v_1, v_2) = $\boldsymbol{v_1} + \boldsymbol{v_2} \in S_i^{(\alpha_1 +_R \alpha_2)}$.

(e) **Negation.** The procedure **neg** computes an encoding of the inverse of a provided element.

- **Input**: **params** - the instance parameters, $\boldsymbol{i} \in \{0, .., k\}$ - the index of the level of encoding, $\boldsymbol{v} \in S_i^{(\alpha)}$ (where $\alpha \in R$) - an encoding of the element to be negated.
- **Output: neg**(**params**, i, v) = $-\boldsymbol{v} \in S_i^{(-_R \alpha)}$.

(f) **Multiplication.** The procedure **mul** computes an encoding of the multiplication of two elements, which may be on different levels of encoding.

- **Input**: **params** - the instance parameters, $0 \leq \boldsymbol{i_1}, \boldsymbol{i_2}$ - the indexes of the encoding levels of the elements (with $i_1 + i_2 \leq k$) , $\boldsymbol{u_1} \in S_{i_1}^{(\alpha_1)}$, $\boldsymbol{u_2} \in S_{i_2}^{(\alpha_2)}$ - the encoding of the elements to be multiplied.
- **Output: mul**(params, i_1, u_1, i_2, u_2) = $\boldsymbol{u_1} \times \boldsymbol{u_2} \in S_{i_1 +_R i_2}^{(\alpha_1 \cdot_R \alpha_2)}$.

(g) **Zero-test.** The procedure **isZero** verifies if the given parameter is a "level-k" encoding of 0.

- **Input**: **params** - the instance parameters, $\boldsymbol{u}$ - a "level-k" encoding of an element in R.
- **Output: isZero**(params, u) = $\mathbf{1}$, if $u \in S_k^{(0)}$, $\mathbf{0}$ otherwise.

(h) **Extraction.** The procedure **ext** realizes the "selection" of a unique "level-k" representative, for every ring element. Also, the elements returned are almost random over $\{0,1\}^\lambda$.

- **Input**: **params** - the instance parameters, $\mathbf{p}_{zt}$ - the zero-test parameter, $\boldsymbol{u}$ - the "level$-k$" encoding of a ring element.
- **Output: ext**(params, $\mathbf{p}_{zt}$, u) = $\boldsymbol{s} \in \{0,1\}^\lambda$, such that:
  (i) $\forall \alpha \in R, v_1, v_2 \in S_k^{(\alpha)}$, it holds that **ext**(params, $\mathbf{p}_{zt}$, v_1) = **ext**(params, $\mathbf{p}_{zt}$, v_2);
  (ii) The distribution $\{$**ext**(params, $\mathbf{p}_{zt}$, v) : $\alpha \in R, v \in S_k^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

Remark 1. *In practice, due to the limitations of computers, the zero-test and the extraction procedures requirements are lessened. Therefore, the zero-test procedure may output 1 for encodings of non-zero elements, with negligible probability in λ. Also, the extraction procedure may output different "level-k" representatives of the same ring element, again with negligible probability in λ.*

Remark 2. *To test if two elements, $u, v \in S_k$, encode the same element $\alpha \in R$, it is sufficient to verify if **isZero**(params, **add**(params, k, u, **neg**(params, k, v)))) returns 1, i.e. $u - v \in S_k^{(0)}$.*

1.4.2 Hardness Assumptions

The current subsection presents the analogues of **MDL** and **MDDH**, discussed in the previous section.

1. **Graded Discrete Logarithm (GDL [11]).** It is said that the GDL problem is hard for a Graded Encoding System GES if for all probabilistic polynomial running time algorithms, the discrete logarithm advantage of an adversary $\mathcal{A}$,

$$\text{AdvDlog}_{GES,\mathcal{A},k}(\lambda) \stackrel{\text{def}}{=} Pr[\mathcal{A}(params,\ \mathbf{p}_{zt},\ u) = a : (params,\ \mathbf{p}_{zt}) \leftarrow \text{InstGen}(1^\lambda, 1^k), \alpha \in R, u \in S_k^{(\alpha)}, a \in S_0^{(\alpha)},$$

   is negligible in λ.

2. **Graded DDH (GDDH [11]).** The GDDH problem is hard for a Graded Encoding System GES if for any probabilistic polynomial running time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the distributions :

$$(\text{params},\ \mathbf{p}_{zt}, a_0, ..., a_k,\ \mathbf{enc}\ (\text{params},\ k, \prod_{i=0}^{k} e_i)) \text{ and}$$
$$(\text{params},\ \mathbf{p}_{zt}, a_0, ..., a_k,\ \mathbf{enc}(\text{params},\ k,\ \mathbf{samp}(\text{params})))$$

   is negligible in λ, where $(params,\ \mathbf{p}_{zt}\ ) \leftarrow \text{InstGen}(1^\lambda, 1^k)$ and for every $i \in \{0, .., k\}$, the referenced elements are: $e_i = \mathbf{samp}(\text{params})$, $a_i = \mathbf{enc}(\text{params},\ 1,\ e_i)$.

   The intuition behind GDDH is that, given $k + 1$ level-one encodings of random elements, one could not easily distinguish a level-k encoding of their product from random.

Chapter 2

Mathematical Background

The purpose of this chapter is to remind the reader basic notions regarding algebra and statistics, but also to analyze in detail concepts and algorithms concerning lattices, especially integer lattices.

2.1 Algebra

The notions of group, cyclic group, ring and polynomial are considered to be previously known by the average reader. More information about the mentioned structures can be found in [14], chapter 2. For a mathematical perspective over groups and polynomial rings properties, and also for a deep incursion in field extension theory, the reader can explore [3].

1. **Ideals and Quotient Rings.**

   Definition 5. *Let $(R, +, \cdot)$ be a commutative finite ring. An **ideal** of R is a nonempty set $\mathcal{I} \subseteq R$ that is closed under addition and $ax = xa \in \mathcal{I}, \forall x \in \mathcal{I}, \forall a \in R$.*

   Let $(R, +, \cdot)$ be a finite ring and $\mathcal{I}$ be an ideal of R. Also, let " $\sim$ " be an equivalence relationship on R, defined by $x \sim y \iff \exists a \in \mathcal{I}$ such that $x = y + a$. The equivalence class of an element $x \in R$ is usually noted with $\hat{x}$, and it represents the set $\{y \in R : y \sim x\}$. The equivalence classes generate a partition of the set R, named *quotient set*, and denoted by $R/\mathcal{I}$. It is known that $(R/\mathcal{I}, +, \cdot)$ is also a ring, and it is called the **quotient ring** $R/\mathcal{I}$.

   Remark 3. *An example of a quotient ring is the well-known ring $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$.*

2. **Cyclotomic polynomials.**

   Definition 6 [3]. *Let $n \geq 1$ be an integer and P_n be the set of all the n^{th} primitive roots of unity. Then, the n^{th} **cyclotomic polynomial** is $\Phi_n = \prod_{\xi \in P_n} (X - \xi)$.*

   Remark 4. Using the fact that $\Phi_{p^k}(X) = \Phi_p(X^{p^{k-1}})$, for any positive integers k, p, with $p-$prime, it can be proved that $\Phi_{2^k}(X) = X^{2^{k-1}} + 1$, for any integer $k \geq 1$.

3. **Vector spaces**.

   Throughout the paper, vectors and matrices are thickened, e.g. $\mathbf{v}$. Also, every vector space is considered to be contained in R^m, with $m \geq 1$, integer.

Definition 7 [14]. *Let m be a positive integer. A **vector space** V is a subset of $\mathbb{R}^m$, such that for every $\alpha_1, \alpha_2 \in \mathbb{R}$ and every $\mathbf{v}_1, \mathbf{v}_2 \in V$,* it holds: $\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 \in V$.

The lecturer is expected to master the concepts of *linear combination, linear independence, basis, vector orthogonality, basis orthogonality.* For a quick review over the mentioned concepts, visit [14].

The only algorithm regarding vector spaces to be presented in the current paper is the **Gram-Schmidt Algorithm**. It receives as input a basis $\{\mathbf{v}_1, .., \mathbf{v}_n\}$ of the vector space V and outputs $\{\mathbf{v}_1^*, .., \mathbf{v}_n^*\}$ - an orthonormal basis of V. The algorithm is presented below:

The Gram-Schmidt Algorithm

1: Set $\mathbf{v}_1^* = \mathbf{v}_1$
2: **for** $i \leftarrow 2$ **to** n **do**
3: Compute $\mu_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j^* \ / \ ||\mathbf{v}_j^*||$, for $1 \leq j < i$
4: Set $\mathbf{v}_i^* = \mathbf{v}_i - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{v}_j^*$
5: **end for**

Intuitively, μ_{ij} represents the length of the projection of $\mathbf{v}_i$ over $\mathbf{v}_j^*$. Therefore, the substraction $\mathbf{v}_i - \mu_{ij}\mathbf{v}_j^*$ generate the projection of $\mathbf{v}_i$ over the orthogonal complement of $\mathbf{v}_j^*$, which leads to the desired output.

2.2 Lattices

2.2.1 General Influence

Lattices have been studied by mathematicians such as Gauss, Lagrange or Minkowski since 18^{th} century, and have been used to prove theorems in number theory and the field extensions. Even though lattices confirmed their significance in mathematics, they were not used in computer science until the 1980s, when Lenstra, Lenstra and Lovàsz proposed the basis-reduction algorithm **LLL**. It represented a major breakthrough in cryptography, and was used to break several cryptosystems, such as RSA (in a low exponent setting) and NTRU.

Since the proposal of **LLL** algorithm, lattices became appealing to the world of cryptography. Thus, since then they have been used in the construction of cryptographic schemes, such as Attribute Based Encryption [6], Fully homomorphic encryption [13] and Graded Encoding Systems [11].

2.2.2 Basic Concepts

Definition 8 ([14]). *Let n, m be two positive integers and let $B = \{\mathbf{v}_1, .., \mathbf{v}_n\} \subset \mathbb{R}^m$ be a set of linearly independent vectors. The **lattice** generated by B is the set:*

$$L \overset{\text{not.}}{=} \mathcal{L}(B) = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + ... + a_n\mathbf{v}_n : a_1, a_2, .., a_n \in \mathbb{Z}\}.$$

Also, a lattice that contains only vectors with integer coordinates is called an ***integer lattice****.* The **dual lattice** is denoted by $L^* = \{\mathbf{y} \in Span(L) : \forall \mathbf{x} \in L, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$.

From a visual point of view, the elements of a lattice are structured as a net, with massive holes between the nodes, as it can be noticed in Figure 1.

Definition 9 ([14]). *Let L be a n-dimensional lattice and $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ be a basis for the lattice L. The* ***fundamental domain*** *for L that is associated with B is:*

$$\mathcal{F}(\mathbf{v}_1, ..., \mathbf{v}_n) = \{t_1\mathbf{v}_1 + ... + t_n\mathbf{v}_n : t_i \in [0, 1), \forall i \in \{1, ..., n\}\}.$$

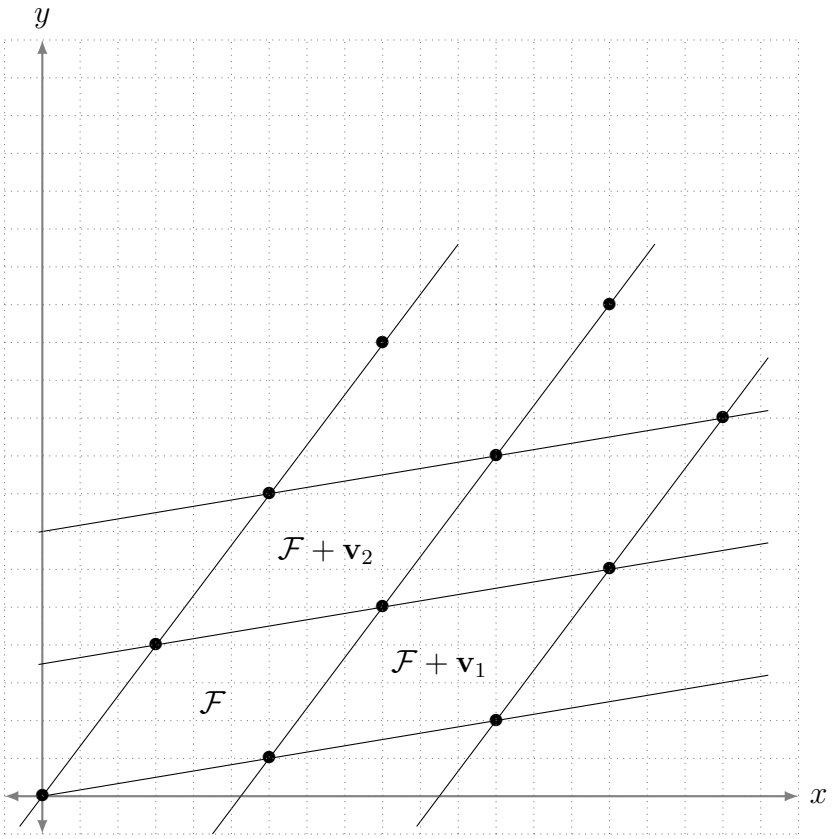

Figure 1: A bidimensional lattice L and the associated fundamental domain $\mathcal{F}$.

Proposition 2. *Let n be a positive integer and $L \subset \mathbb{R}^n$ be a lattice of dimension n. Then, for every basis B of L, the volume of the fundamental domain associated with it is the same.*

Definition 10. *Let n be a positive integer and let $L \subset \mathbb{R}^n$ be a lattice of dimension n. The* ***determinant*** *of L is the volume of any fundamental domain for L, and it is denoted by det(L).*

Definition 11. *Let i be an integer, $i \geq 2$. Also, let n be a positive integer, let $L \subset \mathbb{R}^n$ be a lattice of dimension n and let B be a basis of L. Then, the i^{th}* ***successive minimum*** *is the smallest r such that rB contains i linearly independent lattice points*, and is denoted by:

$$\lambda_i(L) = \min\{r : \dim(\text{span}(L \cap rB)) \geq i\}.$$

Lemma 1 ([19]). Let n be a positive integer, let $L \subset \mathbb{R}^n$ be a lattice of dimension n and let $\epsilon > 0$. Then, the following property holds:

$$\eta_\epsilon(L) \leq \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}} \cdot \lambda_n(L).$$

2.2.3 Hard problems

In order to be able to use lattices in the design of cryptographic schemes, it is required that hard problems related to them to be known. Some of the most important hard problems related to lattices are presented below [14]:

- **Shortest Vector Problem (SVP).** Let n be a positive integer and L be a lattice of dimension n. The problem to find a vector $\mathbf{v} = \underset{\mathbf{w} \in L \setminus \{\mathbf{0}\}}{\operatorname{argmin}} ||\mathbf{w}||$ is referred to as **SVP**.

- **Closest Vector Problem (CVP).** Let n be a positive integer and L be a lattice of dimension n. Also, let $\mathbf{w}$ be a vector in $\mathbb{R}^n \setminus L$. The problem to find a vector $\mathbf{v} \in L$ that satisfies $||\mathbf{w} - \mathbf{v}|| = \min_{\mathbf{u} \in L} ||\mathbf{w} - \mathbf{u}||$ is called **CVP**.

- **Approximate Shortest Vector Problem (apprSVP).** Let $\psi : \mathbb{N} \to \mathbb{R}$ be a function of one parameter and let $\lambda_1(L) \in L$ be one of the shortest vectors in L (i.e. a solution to **SVP**). The problem to find a vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ such that $||\mathbf{v}|| \leq \psi(n) \cdot ||\lambda_1(L)||$ is called **apprSVP**.

- **Approximate Closest Vector Problem (apprCVP).** Let $\psi : \mathbb{N} \to \mathbb{R}$ be a function of one parameter, let $\mathbf{u} \in \mathbb{R}^n \setminus L$ be a non-lattice vector and let $\mathbf{w}$ be a solution to **CVP** associated with $\mathbf{u}$. The problem to find a vector $\mathbf{v} \in L$ such that $||\mathbf{v} - \mathbf{u}|| \leq \psi(n) \cdot ||\mathbf{w} - \mathbf{u}||$ is called **apprCVP**.

Remark 5. ***apprSVP*** *is known to be hard to solve, even for polynomial approximation functions ψ. Also, it is resistant to quantum computing attacks, as opposed to integer factorization problem, which becomes easy in a quantum computing environment, using Schor's algorithm.*

2.2.4 Results concerning short vectors

In order to verify how accurate is the returned solution of an approximation algorithm, it is needed to have an estimate of the desired result. Therefore, an approximative value of the shortest vector length in a lattice is necessary for testing the result of an algorithm solving **apprSVP**.

The current subsection only states the most important results regarding shortest vector length in a lattice. The lecturer who is interested in the proofs of the following results may find them in [14].

Definition 12. *Let n be a positive integer and L be a lattice of dimension n. Also, let $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ be a basis of L. The* ***Hadamard ratio*** *is defined by:*

$$\mathcal{H}(B) = \left(\frac{\det(L)}{||\mathbf{v}_1|| \cdot ||\mathbf{v}_2|| \cdot ... \cdot ||\mathbf{v}_n||} \right)^{1/n}.$$

The Hadamard ratio is a real number in the interval (0, 1] and it represents a measure of the orthogonality of the basis B, with the understanding that the closer the Hadamard ratio is to 1, the more orthogonal are the vectors in B.

Theorem 1 (Minkowski's Theorem [14]). *Let n be a positive integer and $L \subset \mathbb{R}^n$ be a lattice of dimension n. If $S \subset \mathbb{R}^n$ is a symmetric convex set with the property that*

$Vol(S) > 2^n \det(L)$, then S contains a nonzero lattice vector. If S is also a closed set, then it is sufficient to verify that $Vol(S) \geq 2^n \det(L)$.

Theorem 2 (Hermite's Theorem [14]). *Let n be a positive integer. Then, for every lattice L of dimension n, there exists a vector $\mathbf{v} \in L \backslash \{\mathbf{0}\}$ such that $||\mathbf{v}|| \leq \sqrt{n} \cdot \det(L)^{1/n}$.*

Remark 6. *The Hermite's Theorem is a consequence of Minkowski's Theorem, where the set S is considered to be a hypercube in $\mathbb{R}^n$ centered at $\mathbf{0}$. Considering S to be a hypersphere instead of a hypercube, centered at $\mathbf{0}$, the upper bound in Hermite's theorem is lowered by a factor of $\sqrt{\frac{2}{\pi e}}$.*

Proposition 3 ([14]). *Let n be a positive integer and $L \subset \mathbb{R}^n$ be a lattice of dimension n. The **Gaussian heuristic** affirms that the length of the shortest nonzero vector in L is expected to be $\sigma(L) = \sqrt{\frac{n}{2\pi e}} \big(\det(L)\big)^{1/n}$.*

The vigilant reader may note that the *gaussian expected shortest length* is two times smaller than the upper bound presented in *Remark 6*.

2.2.5 LLL Algorithm

The hard problems **SVP** and **CVP** may become easy if an orthogonal basis for the lattice is known in advance. It can be quickly verified that a solution to **SVP** is in fact the shortest vector in the orthogonal basis.

The result is usually accurate even for quasi-orthogonal basis, i.e. basis with a Hadamard ratio reasonably close to 1. Babai designed an algorithm that solves **CVP** in a setting in which a quasi-orthogonal basis is known. **Babai's Algorithm** receives as input a quasi-orthogonal basis $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ of the lattice $L \subset \mathbb{R}^n$ and a vector $\mathbf{w} \in \mathbb{R}^n$. It outputs a vector $\mathbf{v} \in \mathbb{R}^n$, solution to **CVP** problem. The algorithm is presented below, based on [14]:

Babai's Algorithm

1: Find $\alpha_1, \alpha_2, ..., \alpha_n \in \mathbb{R}$ such that $\mathbf{w} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + ... + \alpha_n \mathbf{v}_n$
2: **for** $i \leftarrow 12$ **to** n **do**
3: Set $\beta_i = \lfloor \alpha_i + \frac{1}{2} \rfloor$
4: **end for**
5: $\mathbf{v} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + ... + \beta_n \mathbf{v}_n$

Therefore, solving **SVP** or **CVP** for a lattice L reduces to finding a quasi-orthogonal basis for L. The **LLL** algorithm managed to fill this gap, for lattices of low dimension (i.e. less than 300). Hence, the algorithm had a colossal success among the cryptographers, and it represented the first step to include lattices in the world of cryptography.

Presented in [17], the **LLL** algorithm was initially conceived to provide a polynomial-time algorithm for factoring polynomials with rational coefficients. Two necessary conditions were formulated for a basis $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ in order to be considered *LLL reduced*:

- **Size condition:** $|\mu_{i,j}| = \frac{|\mathbf{v}_i \cdot \mathbf{v}_j^*|}{||\mathbf{v}_j^*||^2} \leq \frac{1}{2}, \forall\ 1 \leq j < i \leq n$;

- **Lovász Condition:** $||\mathbf{v}_i^*|| \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)||\mathbf{v}_{i-1}^*||^2, \forall\ 1 < i \leq n$,

where $B^* = \{\mathbf{v}_1^*, ..., \mathbf{v}_n^*\}$ is the orthogonal basis returned by the **Gram-Schmidt** algorithm and $\mu_{i,j}$ refers to the constants defined in the same algorithm.

Proposition 4. (LLL reduced basis apprSVP [14]). *Let n be a positive integer and let $L \subset \mathbb{R}^n$ be a lattice of dimension n. For any **LLL reduced basis** $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$, the following property holds:*

$$||\mathbf{v}_1|| \leq 2^{(n-1)/2} \cdot \min_{\mathbf{v} \in L \setminus \{\mathbf{0}\}} ||v||.$$

As it can be easily observed, using the **LLL** lattice-reduction algorithm leads to solving **apprSVP** by a factor of $2^{(n-1)/2}$.

LLL algorithm is presented below, in the version illustrated by the book of Hoffstein, Pipher and Silverman, [14]. The input of the algorithm is a basis $B = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$, and the output is the basis B, modified as a *LLL reduced basis*:

LLL Algorithm

Set k = 2
Set $\mathbf{v}_1^* = \mathbf{v}_1$
while $k \leq n$ **do**
    for $j \leftarrow k - 1$ **downto** 1 **do**
        Set $\mathbf{v}_k = \mathbf{v}_k - \lfloor \mu_{k,j} + \frac{1}{2} \rfloor \mathbf{v}_j$ ▷ **Size Reduction**
    end for
    if $||\mathbf{v}_k^*||^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right)||\mathbf{v}_{k-1}^*||^2$ **then** ▷ **Lovász Condition**
        Set $k = k + 1$
    else
        Swap $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$
        Set $k = \max(k - 1, 2)$
    end if
end while

Theorem 3 (LLL Correctness and Running Time [14]). *Let n be a positive integer, let $L \subset \mathbb{R}^n$ be a lattice of dimension n and let $B = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ be a basis for L. Then, the **LLL algorithm**, presented above, returns an **LLL reduced basis** for L, and also it terminates in a finite number of steps.*

Remark 7. *The algorithm executes the steps [3]-[13] no more than $\mathcal{O}(n^2 \log n + n^2 \log D)$ times, where $D = \max_{\mathbf{w} \in B} ||\mathbf{w}||$. Thus, LLL is a polynomial-time algorithm.*

LLL proved its utility in cryptanalysis, its applications covering attacks on the family of knapsack public-key cryptosystems, but also on GGH and NTRU cryptosystems.

2.2.6 Ideal Lattices

Let k be a positive integer and let $n = 2^k$. Using *remark 4*, subsection *cyclotomic polynomials*, it can be easily proved that $\Phi_{2n}(X) = \Phi_{2^{k+1}}(X) = X^{2^k} + 1 = X^n + 1$. Let $R = \mathbb{Z}[X]/(X^n+1)$ be the $(2n)^{th}$ cyclotomic polynomial ring and let $R_q = \mathbb{Z}_q[X]/(X^n+1)$.

One of the possible representations of the elements in R is using its vector of coefficients. For example, a polynomial $P(X) = a_{n-1}X^{n-1}+...+a_1X+a_0 \in R$ may be uniquely identified by the vector of coefficients $(a_{n-1}, ..., a_1, a_0) \in \mathbb{Z}^n$. Addition of polynomials in the mentioned representation is realized component-wise, while multiplication requires a bit more complicated rule, that should output the desired result.

Let $g \in R$ be an element of R. The **principal ideal** in R generated by g is denoted by $\langle g \rangle = \{g \cdot x : x \in R\}$.

Definition 13 (Ideal lattice). *Let n be a power of two and let $R = \mathbb{Z}[X]/(X^n + 1)$ be the $(2n)^{th}$ cyclotomic polynomial ring. Also, let $g \in R$ be an element of R. Then, the principal ideal generated by g is called to be an **ideal lattice**, accentuating the duality of the structure: it is at the same time an ideal and a lattice.*

Let $g \in R$ be an element of R and let $B(G) = \{g, gX, ..., gX^{n-1}\}$ be a basis of $\langle g \rangle$. Also, let $v \in R$ be an arbitrary element in R. The **reduction modulo the fundamental domain** of $B(g)$ is denoted by $[u]_g$, and it represents the unique $v' \in R$ such that $v - v' \in \langle g \rangle$ and $v' = \sum_{i=0}^{n-1} \alpha_i X^i g$, with $\alpha_i \in [-\frac{1}{2}, \frac{1}{2})$.

Proposition 5 ([18]). Let n, m be positive integers, with $m \geq 2$ and $n = \varphi(m)$. Also, let $\mathcal{I}$ be an arbitrary ideal of the m^{th} cyclotomic ring. Then, $\lambda_n(\mathcal{I}) = \lambda_1(\mathcal{I})$.

2.3 Probabilities and Statistics

The construction of the Graded Encoding Scheme exposed in [11] requires in-depth results concerning probabilities and statistics, mainly due to the nondeterministic character of the encodings of elements. Therefore, the current section presents the essential results regarding discrete Gaussian distributions over lattices, the sum of discrete Gaussians and the smoothing parameter for a lattice.

1. **Gaussian distributions [1].** The ellipsoid, continuous $n-$ dimensional Gaussian distribution, with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is denoted by $\mathcal{N}^n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and has the density function $f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-\mathbf{1}}(\mathbf{x}-\boldsymbol{\mu})\big)$.

   Definition 14. *Let m, n be two positive integers, let $S \in \mathbb{R}^{m \times n}$ be a rank-n matrix and let $\boldsymbol{\mu} \in \mathbb{R}^n$ be an $n-$dimensional vector. The **ellipsoid Gaussian function** over $\mathbb{R}^n$, centered at $\boldsymbol{\mu}$ and with parameter S is denoted by:*

$$\rho_{S,\boldsymbol{\mu}}(\mathbf{x}) = \exp\big(-\pi(\mathbf{x}-\boldsymbol{\mu})^T(S^TS)^{-1}(\mathbf{x}-\boldsymbol{\mu})\big), \forall \mathbf{x} \in \mathbb{R}^n.$$

   For the particular case $\boldsymbol{\mu} = \mathbf{0}$, the short version $\rho_S(\cdot)$ is used. Also, the **spherical** case is achieved when $S = \sigma I_n$, with $\sigma \in \mathbb{R}^*$.

Definition 15. *Let m, n be two positive integers, let $S \in \mathbb{R}^{m \times n}$ be a rank-n matrix and let $L \subset \mathbb{R}^n$ be an $n-$dimensional lattice. The* ***ellipsoid discrete Gaussian distribution*** *over L, centered at* ***0*** *and with parameter S is:*

$$\mathcal{D}_{L,S}(\mathbf{x}) = \frac{\rho_S(\mathbf{x})}{\rho_S(L)}, \forall \mathbf{x} \in L,$$

where $\rho_S(L) = \sum_{\mathbf{x} \in L} \rho_S(\mathbf{x})$.

2. **Smoothing parameter [19].** Intuitively, the *smoothing parameter* of a lattice represents a lower bound for the value of the radius of a discrete Gaussian distribution $\mathcal{D}$ with the property: if a noise vector is extracted from $\mathcal{D}$ and is reduced modulo the fundamental domain of the lattice, then the resulted distribution is close to uniform. Formally, the smoothing parameter is defined below:

   Definition 16 (Smoothing parameter [19]). Let n be a positive integer, let $L \subset \mathbb{R}^n$ be an $n-$dimensional lattice and let ϵ be a positive real number. The **smoothing parameter** for L and ϵ is denoted by $\eta_\epsilon(L)$ and represents the smallest $s \in \mathbb{R}$ such that $\rho_{1/s}(L^* \backslash \{\mathbf{0}\}) \leq \epsilon$.

   Lemma 2 ([1]). Let m, n be two positive integers, let $L \subset \mathbb{R}^n$ be an $n-$dimensional lattice, $\epsilon \in (0, 1)$ and let $S \in \mathbb{R}^{m \times n}$ be a rank-n matrix such that $\sigma_n(S) \geq \eta_\epsilon(L)$. Then,

$$\Pr_{v \leftarrow \mathcal{D}_{L,S}} \left(||\mathbf{v}|| \geq \sigma_1(S)\sqrt{n} \right) \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n},$$

   where $\sigma_1(S), \sigma_n(S)$ denote the largest, respectively the least singular values of S.

3. **Sum of discrete gaussians.**
   Theorem 4 ([11]). *Let n be a positive integer, let $L \subset \mathbb{R}^n$ be a lattice of dimension n and let B be a matrix whose rows form a basis of L. Also, let $w = \frac{\sigma_1(B)}{\sigma_n(B)}$, let ϵ be a constant negligible in n, and let m, s, s' be parameters such that $s \geq \eta_\epsilon(\mathbb{Z}^n), m \geq 10n \log(8(mn)^{1.5}sw)$ and $s' \geq 4mnw \ln \frac{1}{\epsilon}$.*
   Then, when choosing the rows of an $m \times n$ matrix X from the ***spherical Gaussian*** *over L, $X \leftarrow (\mathcal{D}_{L,s})^m$, it is true with all but probability $2^{-O(m)}$ over the choice of X that the* ***statistical distance*** *between $\varepsilon_{X,s'}$ and the* ***ellipsoid Gaussian*** *$\mathcal{D}_{L,s'X}$ is bounded by 2ϵ. Here, $\varepsilon_{X,s'} = \{X^T\mathbf{v} : \mathbf{v} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}\}$.*

Chapter 3

Proposed Encoding Scheme

Garg, Gentry and Halevi constructed a Graded Encoding Scheme based on ideal lattices, in [11]. Their construction was the first candidate to approximate multilinear maps, therefore had a tremendous impact in the world of cryptography. The current chapter intends to present the system designed by the three aforementioned authors, in an in-depth manner.

3.1 Overview

To start with, the ring R denotes the cyclotomic polynomial ring $\mathbb{Z}[X]/(X^n+1)$, for some integer n - power of two. The ring R is often considered to be the lattice $\mathbb{Z}^n$, because a correspondence between the two structures is obvious. Also, $g \in R$ is a short ring element and $\mathcal{I} = \langle g \rangle$ is the principal ideal generated by g. Additionaly, an element $\mathbf{z}$ is extracted randomly in R_q.

The elements to be encoded by the scheme are the equivalence classes (or **cosets**) of the quotient ring $QR = R/\mathcal{I}$, denoted by $\hat{e}$, for some $e \in R$.

A level-zero encoding of a coset $\hat{e}$ is a short vector in that coset. The existence of a small vector in any coset is assured by the fact that g is a short element, therefore the basis $B(g) = \{g, Xg, ..., X^{n-1}g\}$ has all elements small - only circular permutations of the vector g, eventually with a changed sign. Hence, the fundamental domain itself of $B(g)$ is small and because for any $e \in R$, there exists an element $e_g \in \mathcal{F}(B(g))$ such that $e_g \in \hat{e}$, the result follows immediately.

For any $i \in \{1, 2..., k\}$, the set of all level-i encodings of a coset $\hat{e}$ is $S_i^{\hat{e}} = \left\{ \frac{c}{\mathbf{z}^i} \in R_q : c \in eh, ||c|| < q^{1/8} \right\}$. The value $||c||$ will be referred to as the **noise level** of the encoding.

3.2 Efficient Procedures

The procedures to be presented are a specific case of the efficient procedures introduced in Subsection 1.4.1. Thus, in this section only the implementation of the mentioned functions will be exposed, as in [11], with a high-level algorithm exposed in the beginning of every procedure, and the in-depth explanation afterwards. Note that R may be identified by $\mathbb{Z}^n$, and R_q by $\mathbb{Z}_q^n$. Also, K denotes the field $\mathbb{Q}[X]/(X^n+1)$.

(a) **Instance generation**: (params, $\mathbf{p}_{zt}$) $\leftarrow$ InstGen($1^\lambda, 1^k$).

InstGen($1^\lambda, 1^k$)

1: Choose $g \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ (encoding of the generator, element kept **private**).

2: Choose $\mathbf{z} \leftarrow R_q$ (encoding of a **private** element, used for encoding elements on level higher than 0).

3: Choose $a \leftarrow \mathcal{D}_{\hat{1},\sigma''}$, then compute a **public** level-one representation of $\hat{1}$, $\mathbf{y} = \left[\frac{\mathbf{a}}{\mathbf{z}}\right]_q$.

4: Choose $\mathbf{b}_i \leftarrow \mathcal{D}_{\hat{0},\sigma'''}, \forall i \in \{1, 2, ..., m\}$ and compute $\mathbf{x}_i = \left[\frac{\mathbf{b}_i}{\mathbf{z}}\right]_q, \forall i \in \{1, 2, ..., m\}$, **public** level-one representations of $\hat{0}$. Also, set $X = (\mathbf{x}_1|\mathbf{x}_2|...|\mathbf{x}_m)^T$.

5: Choose $\mathbf{h} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sqrt{q}}$ such that $h \notin \mathcal{I}$, then compute the zero-testing parameter $\mathbf{p}_{zt} = \left[\frac{\mathbf{h}\mathbf{z}^k}{\mathbf{g}}\right]_q$.

6: Choose $s \leftarrow \mathbb{Z}$.

7: **Output**: params $= (n, q, \mathbf{y}, \mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_m, s)$, $\mathbf{p}_{zt}$.

1: The uniform extraction of g from the spherical Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n,\sigma}$, with $\sigma = \sqrt{\lambda n}$, is performed repeatedly, until the following conditions are met:
- $||g|| \leq \sigma\sqrt{n}$ and g is invertible in R_q;
- g^{-1} is a short vector;
- $\mathbf{N}(g)$ is a prime $\geq 2^{O(n)}$, where $\mathbf{N}(g)$ denotes the norm of the ideal $\langle g \rangle$, as defined in [10], chapter 5.

The last condition implies that $\mathcal{I} = \langle g \rangle$ is a **prime ideal**. Also, **Lemma 6.1** from [10] proves that the algorithm above completes in polinomially many trials.

2: $\mathbf{z}$ is chosen uniformly from R_q, thus with overwhelming probability is not "small". Using [10] (*Lemma 5.20*), it follows that with overwhelming probability, $\mathbf{z}$ is invertible in R_q.

3: Parameters $\mathbf{y}$ and X will show their usefulness during the **higher-level encoding** procedure. Also, $\mathbf{p}_{zt}$ is used as a zero-testing parameter, while s is used as a seed for a strong randomness extractor.

(b) **Sampling level-zero encodings**: $d \leftarrow$ **samp**(params).

samp(params)

1: Choose $d \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma'}$.

2: **Output**: $\mathbf{d}$.

It can be proved that, if $\sigma' \geq \sqrt{\lambda n}||g||$, then the distribution ($\mathbf{d} \mod \mathcal{I}$) is statistically close to the uniform distribution over ($\mathbb{Z}^n \mod \mathcal{I}$). Therefore, the coset to be sampled is close to uniform.

To prove it, firstly it can be stated that, because $g \in \mathcal{I}$, then $\lambda_1(\mathcal{I}) \leq ||g||$. Also, knowing that $\mathcal{I}$ is an ideal of the $(2n)^{th}$ cyclotomic ring, with n a power of two, it follows from **Proposition 5** that $\lambda_n(L) = \lambda_1(L)$. Corroborating the two results, it results that $\lambda_n(L) \leq ||g||$. Therefore, applying **Lemma 1**, it follows that:

$$\eta_{2^{-\lambda}}(\mathcal{I}) \leq \sqrt{\frac{\ln(2n(1+2^\lambda))}{\pi}} \cdot \lambda_n(L) \leq \eta_{2^{-\lambda}}(\mathcal{I}) \leq \sqrt{\frac{\ln(2n(1+2^\lambda))}{\pi}} \cdot ||g|| \leq \sqrt{n\lambda}||g||.$$

Since $\sigma' \geq \sqrt{\lambda n}||g||$, it is obvious that $\sigma' \geq \eta_{2^{-\lambda}}(\mathcal{I})$, therefore the proof is complete.

Also, the size of $\mathbf{d}$ is bounded by $\sigma'\sqrt{n}$, result proved by **Lemma 2**.

(c) **Encodings at higher levels:** $\mathbf{u}_i \leftarrow$ enc(params,$i, \mathbf{d}$).

enc(params,$i, \mathbf{d}$)

1: **if** $i == 1$ **then**
2: Choose $\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma^*}$.
3: Set $\mathbf{u}_i = \left[\frac{\mathbf{yd}}{\mathbf{z}} + X^T\mathbf{r}\right]_q$.
4: **else**
5: Set $\mathbf{u}_i = [\mathbf{d} \cdot \mathbf{y}^i]_q$.
6: **Output:** $\mathbf{u}_i$.
7: **end if**

The necessity of publishing a level-one representation of an element is obvious, from the inherent structure of the construction. The "lifting" of a representation, from a lower level to a higher one, may be performed only using another representation of an element, on a higher level.

In order to preserve the element to be encoded and to change only the level of encoding, the public element $\mathbf{y}$ is an encryption of element $\hat{1}$ - the neutral element at multiplication in QR. Also, because $\mathbf{y}$ is a level-one encoding, multiplying another element by it results in "lifting" the representation only one level. Therefore, combining the two arguments, it follows that multiplying any level-i representation of an element $\alpha \in QR$ by $\mathbf{y}$, it results a level-$(i+1)$ representation of α.

More formally, given a level-zero encoding $\mathbf{d}$, then the level-one representation of $\mathbf{d}$ is $\mathbf{u}_1 = [\mathbf{dy}]_q = \left[\frac{\mathbf{da}}{\mathbf{z}}\right]$, where $a \leftarrow \mathcal{D}_{\hat{1},\sigma''}$, therefore an element $\mathbf{x} \in \mathcal{I}$ exists such that $a = 1 + \mathbf{x}$. Then, $\mathbf{d}a = \mathbf{d} + \mathbf{dx}$, with $\mathbf{x} \in \mathcal{I}$, so $\mathbf{d}a \in \hat{\mathbf{d}}$. Also, $||\mathbf{d}a|| \leq ||d|| \cdot ||a|| \cdot \sqrt{n}$, which is a polynomial in n. Thus, $\mathbf{u}_1 = [\mathbf{dy}]_q$ is a valid level-one encoding of $\mathbf{d}$.

Generally, in order to generate a level-i encoding of $\mathbf{d}$, one can simply multiply $\mathbf{d}$ by $\mathbf{y}^i$, and get: $\mathbf{u}_i = [\mathbf{dy}^i]_q = \left[\frac{\mathbf{d}a^i}{\mathbf{z}^i}\right]$. It can be seen that, for a similar reason as above, $\mathbf{d}a^i \in \hat{\mathbf{d}}$ and also $||\mathbf{d}a^i|| \leq ||\mathbf{d}|| \cdot ||a|| \cdot n^{i/2}$, therefore $\mathbf{u}_i$ is a valid level-i encoding of $\mathbf{d}$.

Usually, the applications of Graded Encoding System require that a level-one representation of an element to be made public, without the possibility to recover a level-zero representation of that element. Thus, the process of generating $\mathbf{u}_1$ requires a more extensive technique. That is because in the setting above, one can easily recover $\mathbf{d}$ by simply dividing $\mathbf{u}_1$ to $\mathbf{y}$ in R_q, which is unacceptable.

Therefore, every level-one encoding of an element is randomized, in order to prevent easy recovery of plain-text. The procedure is as follows: given $\mathbf{d}$, a level-zero representation of an element, the first step is to extract an element $\mathbf{r}$ from the spherical discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m,\sigma^*}$, for a large enough σ^*. Then, the result is:

$$\mathbf{u}_1 = [\mathbf{y}\mathbf{d} + X^T\mathbf{r}]_q = \left[\mathbf{y}\mathbf{d} + \sum_{i=1}^{m} \mathbf{x}_i \cdot r_i\right]_q = \left[\frac{a\mathbf{d} + \sum_{i=1}^{m} r_i\mathbf{b}_i}{\mathbf{z}}\right]_q.$$

Knowing that $b_i \in \hat{0}, \forall i \in \{1, 2, .., m\}$ and that $a \in \hat{1}$, it easily follows that $a\mathbf{d} + \sum_{i=1}^{m} r_i\mathbf{b}_i \in \hat{\mathbf{d}}$. Also, the length of the numerator is upper-bounded by $\sigma^* \cdot \text{poly}(m, n)$. Also, if the parameters $\mathbf{b}_i$ are chosen from a fairly wide enough spherical Gaussian distribution, then from **Theorem 4** it follows that the distribution of $B\mathbf{r}$ is close to a wide ellipsoidal Gaussian. Hence, the distribution of $a\mathbf{d} + B\mathbf{r}$ is nearly independent of $a\mathbf{d}$, so the encoding is truly randomized.

Remark 8. *Randomization may be performed at any level, not only the first one. Besides, other versions of Graded Encoding Schemes(e.g. [9]) include a procedure for randomization at level-one, denoted* ***reRand****. The current setting is more restrictive, but has the advantage that it realizes a randomization for every level-one encoding of an element, reducing the implementation errors due to negligence.*

(d) **Adding encodings: add**(params, i, v_1, v_2)

add(params,i, v_1, v_2)

1: **Output:** $[v_1 + v_2]_q$.

The addition in the current settings is a very simple operation, as it can be observed from the one-lined algorithm presented above. To verify the correctness of the addition algorithm, one may consider v_1, v_2, two level$-i$ encodings of the cosets $\widehat{e_1}$, respectively $\widehat{e_2}$. Therefore, there exist two elements $c_1 \in \widehat{e_1}, c_2 \in \widehat{e_2}$ such that $v_1 = \left[\frac{c_1}{\mathbf{z}}\right]_q$ and $v_2 = \left[\frac{c_2}{\mathbf{z}}\right]_q$, and c_1, c_2 are short vectors. Adding the two encodings yields:

$$[v_1 + v_2]_q = \left[\left[\frac{c_1}{\mathbf{z}}\right]_q + \left[\frac{c_2}{\mathbf{z}}\right]_q\right]_q = \left[\frac{c_1+c_2}{\mathbf{z}}\right]_q,$$

with $c_1 + c_2 \in \widehat{e_1 + e_2}$ and $c_1 + c_2$ a short vector in R. Thus, the exposed addition operation is correct.

Remark 9. *Addition operation can be extended to support the summation of many elements simultaneously. It can be effortlessly seen that the size of the numerator*

remains small, even though addition is performed many times (e.g. a number of times polynomial in n).

(e) **Multiplying encodings: mul**(params, v_i, v_j).

mul(params, v_i, v_j)

1: **Output:** $[v_i \cdot v_j]_q$.

Multiplication operation is as simple as the addition one. The proof of correctness follows immediately.
Let v_i be a level$-i$ encoding of a coset $\widehat{e_1}$ and let v_j be a level$-j$ encoding of a coset $\widehat{e_2}$, with $0 \leq i, j$ and $i + j \leq k$. Then, using the properties of the construction, one may note that there must exist two vectors, $c_1 \in \widehat{e_1}, c_2 \in \widehat{e_2}$, such that $v_i = \left[\frac{c_1}{\mathbf{z}^i}\right]_q$ and $v_j = \left[\frac{c_2}{\mathbf{z}^j}\right]_q$. Performing the multiplication of the encodings, as presented in the algorithm, the result is:

$$[v_i \cdot v_j]_q = \left[\left[\frac{c_1}{\mathbf{z}^i}\right]_q \cdot \left[\frac{c_2}{\mathbf{z}^j}\right]_q\right]_q = \left[\frac{c_1 \cdot c_2}{\mathbf{z}^{i+j}}\right]_q,$$

with $c_1 \cdot c_2 \in \widehat{e_1 \cdot e_2}$, and $c_1 \cdot c_2$ a short vector in R (thus unchanged in R_q). Hence, the multiplication operation exposed in the algorithm is correct.

Remark 10. *Usually, the applications of Graded Encoding Systems will require to multiply k level-one encodings, thus the property stated at* ***Remark 8*** *also holds true for multiplication operation. As a result, it can be easily noted that the product of k level-one encodings represents a level$-k$ encryption of an element.*

(f) **Zero testing:** isZero(params, $\mathbf{p}_{zt}$, u)

isZero(params, $\mathbf{p}_{zt}$, u)

1: **if** $||\mathbf{p}_{zt} \cdot u||_\infty \leq q^{\frac{3}{4}}$ **then**
2: Set $result = 1$
3: **else**
4: Set $result = 0$
5: **end if**
6: **Output:** $result$

In order to test if two encodings represent the same element (obviously, on the same level), it is sufficient to substract them and compare to zero. The zero testing procedure involves a so-called zero-testing parameter and may be used only for level$-k$ encodings. However, despite the fact that the procedure always returns the correct answer for encodings of zero, it also may return (with negligible probability) a false answer for a non-zero encoding.

As presented in the instance generation algorithm, the zero-testing parameter is $\mathbf{p}_{zt} = \left[\frac{\mathbf{h}\mathbf{z}^k}{\mathbf{g}}\right]_q$, where $\mathbf{h} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sqrt{q}}$. Also, from the inherent structure of the construction it follows that there exists a short element c, such that $u = \left[\frac{c}{\mathbf{z}^k}\right]_q$. To prove the *general* faultlessness of the algorithm, it is required to remark that:

$$w = \mathbf{p}_{zt} \cdot u = \left[\frac{\mathbf{h}\mathbf{z}^k}{\mathbf{g}}\right]_q \cdot \left[\frac{c}{\mathbf{z}^k}\right]_q = \left[\frac{\mathbf{h} \cdot c}{\mathbf{g}}\right]_q.$$

- If u **is an encoding of zero**, then $c \in \hat{0}$, therefore $c \in \mathcal{I}$. Because $\mathcal{I} = \langle g \rangle$, it follows that $c = g \cdot \alpha$, for $\alpha \in R$. Thus, the element $\frac{c}{\mathbf{g}} \in R$ is the same element as $c \cdot \mathbf{g}^{-1} \in \mathbb{K}$. It follows that $||w|| \leq ||\mathbf{h}|| \cdot ||c|| \cdot ||\mathbf{g}^{-1}|| \cdot \sqrt{n}^2$. Then, using **Theorem 3** for the case of choosing $\mathbf{h} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sqrt{q}}$, it can be easily noticed that $||\mathbf{h}|| \leq \sqrt{q} \cdot \sqrt{n}$. Corroborating the last two results with the fact that $||\mathbf{g}||$ is small (validated by the generation algorithm), it follows that:

$$||w|| \leq q^{\frac{1}{2}} \cdot \sqrt{n} \cdot q^{\frac{1}{8}} \cdot ||\mathbf{g}^{-1}|| \cdot n \leq q^{\frac{3}{4}}.$$

- If u **is an encoding of a non-zero coset**, then $c \notin \hat{0}$. In order to prove that the value $||w||$ is larger than $q^{\frac{3}{4}}$ with overwhelming probability, one more result must be settled below.

  Lemma 3 (Lemma 6.5, [10]). *Let $w = \left[\frac{c\boldsymbol{h}}{\boldsymbol{g}}\right]_q$, with the properties: $||\boldsymbol{g}w|| \leq \frac{q}{2}$ and $||c\boldsymbol{h}|| \leq \frac{q}{2}$. If $\langle \boldsymbol{g} \rangle$ is a prime ideal, then either c or $\boldsymbol{h}$ is in the ideal $\boldsymbol{g}$.*

  Using the same reason as in the first situation, it can be easily seen that $||\mathbf{h}|| \leq \sqrt{q} \cdot \sqrt{n}$, with overwhelming probability. Then, because $||c|| \leq q^{\frac{1}{8}}$ (from the intrinsic structure of the construction), it follows that:

  $$||c \cdot \mathbf{h}|| \leq \sqrt{n} \cdot q^{\frac{1}{2}} \cdot q^{\frac{1}{8}} \cdot \sqrt{n} = q^{\frac{5}{8}} \cdot n < \frac{q}{2}. \quad (1)$$

  Supposing that there exists $\epsilon > 0$ such that $||w|| \leq q^{1-\epsilon}$, then $||w\mathbf{g}|| < q^{1-\epsilon} \cdot ||\mathbf{g}|| \cdot \sqrt{n}$ and because $||g||$ is polynomial in n, it directly results that:

  $$||w\mathbf{g}|| < \frac{q}{2}. \quad (2)$$

  Using the results (1) and (2) in collaboration with **Lemma 3**, it follows that either c or $\mathbf{h}$ is in the ideal $\langle \mathbf{g} \rangle$, which is a **contradiction**, because $c \notin \hat{0}$ and $\mathbf{h} \notin \mathcal{I}$. Then, for any $\epsilon > 0$, $||w|| \geq q^{1-\epsilon}$, therefore $||w|| \geq q^{\frac{3}{4}}$.

Remark 11. *The zero testing algorithm could have been written in a more condensed (and elegant) manner. However, in order to keep the same structure of the algorithm throughout the chapter, the current version was adopted.*

(g) **Extraction:** $r \leftarrow \text{ext}(\text{params}, \mathbf{p}_{zt}, u)$

ext(params, $\mathbf{p}_{zt}$, u)

1: Set $M = \mathbf{msbs}_{\frac{\log q}{4} - \lambda}\left([u \cdot \mathbf{p}_{zt}]_q\right)$

2: **Output:** $\text{EXTRACT}_s(M)$

The extraction procedure is fairly simple. First, given u, a valid level$-k$ representation of a coset, multiply it with $\mathbf{p}_{zt}$ (in R_q). Then extract the most significant

$\frac{\log q}{4} - \lambda$ bits of *each* of the coefficients of the result. Finally, apply a strong randomness extractor on the resulted bits. The algorithm was divided in two steps in order to stress its dual importance.

The first step realizes a *canonicalization* of cosets representation (the extraction procedure, applied to two different encodings of the same coset, must return the same result). On the other side, the second step performs a *randomization* of the result, so that an external observer will not be capable to generate a valid standard representation of a desired coset.

The correctness of the first step is further discussed. Let $\widehat{e_1}, \widehat{e_2} \in R/\mathcal{I}$ be two cosets, let $u_1 = \left[\frac{c_1}{\mathbf{z}^k}\right]_q$ and let $u_2 = \left[\frac{c_2}{\mathbf{z}^k}\right]_q$, with $c_1 \in \widehat{e_1}, c_2 \in \widehat{e_2}$.

- If $\widehat{e_1} = \widehat{e_2}$, then u_1 and u_2 are **encodings of the same coset**. Then, using the results from *zero-testing procedure*, it follows that $||\mathbf{p}_{zt}u_1 - \mathbf{p}_{zt}u_2|| = ||\ \mathbf{p}_{zt}(u_1 - u_2)|| \leq q^{\frac{3}{4}}$. Therefore, with overwhelming probability, the most significant $\frac{\log q}{4}$ bits of every coefficient of $(\mathbf{p}_{zt}u_1 - \mathbf{p}_{zt}u_2)$ are equal to 0, hence $\mathbf{p}_{zt}u_1$ and $\mathbf{p}_{zt}u_2$ have the same most significant $(\frac{\log q}{4} - \lambda)$ bits for each coefficient. So, the same result is returned by the procedure for u_1 and u_2.
- If $\widehat{e_1} \neq \widehat{e_2}$, then u_1 and u_2 are **encodings of different cosets**. Therefore, it is known from *zero-test procedure* that for every $\epsilon > 0$, $||\mathbf{p}_{zt}(u_1 - u_2)|| \geq q^{1-\epsilon}$. It is now obvious that $\mathbf{p}_{zt}u_1$ and $\mathbf{p}_{zt}u_2$ do not have the same most significant $(\frac{\log q}{4} - \lambda)$ bits for each coefficient, hence the procedure returns different results for u_1 and u_2.

Remark 12. *The element given as parameter to the Extract procedure must be a valid level−k representation of a coset in $R/\mathcal{I}$. This information is used within the proof of correctness for the procedure, hence on other levels of encoding, the behavior is undefined.*

3.3 Parameter constraints

All of the above procedures rely on appropriate settings for the parameters, especially regarding the width of the Gaussian distributions. The constraints will be explained in-depth for every choice of parameter value.

- $\boldsymbol{\sigma = \sqrt{\lambda n}}$. The parameter σ defines the width of the first involved Gaussian, used for extraction of the generator $\mathbf{g}$. The parameter must satisfy $\sigma \geq \eta_{2^{-\lambda}}$, so setting $\sigma = \sqrt{\lambda n}$ is safe. Note that, using **Theorem 3**, $||g|| \leq \sigma\sqrt{n} = n\sqrt{\lambda}$, with overwhelming probability.
- $\boldsymbol{\sigma' = \lambda n^{\frac{3}{2}}}$. In the paragraphs reserved for *sampling* procedure, it was explained why σ' should be lower bounded by $||g|| \cdot \sqrt{\lambda n}$. Using the upper bound for $||g||$, it follows that $\sigma' = \lambda n^{\frac{3}{2}}$ is a good parameter choice.
- $\boldsymbol{\sigma^* = 2^{\lambda}}$. Parameter σ^* defines the width of the Gaussian distribution used to draw the randomization vector $\mathbf{r}$ from *higher level encoding* procedure. In order to apply **Theorem 4**, σ^* must satisfy: $\sigma^* > poly(n, m, \lambda)$. Also, for the numerator of the randomized encoding to become independent of the numerator before the

randomization, it suffices that $\sigma^* = 2^\lambda$. With this value for σ^*, a level-one encoding of a coset takes the form $\left[\frac{c}{\mathbf{z}}\right]_q$, with $||c|| \leq 2^\lambda \cdot poly(n)$.

- $\boldsymbol{q = 2^{8k\lambda} \cdot n^{O(k)}}$. Usually, applications of *Graded Encoding Schemes* compute a level$-k$ encoding of a coset by multiplying k level-one representations of cosets. Using the result above regarding the upper bounds for numerators of level-one encodings, it follows that the numerator of a level$-k$ representation will be upper bounded by: $(2^\lambda \cdot poly(n))^k$. Thus, in order for the product to be considered a valid encoding (which means that the size of numerator should not exceed $q^{\frac{1}{8}}$), it is necessary that: $(2^\lambda \cdot poly(n))^k \leq q^{\frac{1}{8}}$. Finally, by setting $q = 2^{8k\lambda} \cdot n^{O(k)}$ it is ensured that the value of q is big enough for all the operations to be performed in a safe manner.

- $\boldsymbol{n > O(k\lambda^2)}$. For the scheme to assure a security level of λ, it is needed that $q < 2^{\frac{n}{\lambda}}$. Therefore, using the above restriction for q, it immediately results that $n > O(k\lambda^2)$.

- $\boldsymbol{m = O(n^2)}$. In order to apply **Theorem 4** during *higher level encodings* proof of correctness, it is needed that $m > n \log q$. From the previous bounds for q and n, it yields that $m = O(n^2)$.

3.4 Hardness Assumptions

This section is linked directly to **Subsection 1.4.2** by presenting the **GDL** and **GDDH** problems and their implications in the current settings.

1. **Graded Discrete Logarithm (GDL).** *It is said that the GDL problem is hard for a Graded Encoding System GES if for all probabilistic polynomial running time algorithms, the discrete logarithm advantage of an adversary $\mathcal{A}$,*

$$\text{AdvDlog}_{GES,\mathcal{A},k}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathcal{A}(params,\, \mathbf{p}_{zt},\, u) = a : (params,\, \mathbf{p}_{zt}) \leftarrow \text{InstGen}(1^\lambda, 1^k), c \in R, u = \left[\frac{c}{\mathbf{z}^k}\right]_q, a \in \hat{c},$$

is negligible in λ.

Remark 13. *The vigilant reader may note that the **GDL** problem is only defined for level$-k$ encodings of cosets. That is because if the element u is a valid level$-i$ representation, with $1 \leq i < k$, the problem may become weak, as explained in [11], Section 4.4.*

In order to introduce the **GDDH** problem, it is required to define an algorithm that generates a **GDDH** instance, as the one presented in [10], Section 7.1:

genGDDH(params, $\mathbf{p}_{zt}$)

Choose $\mathbf{f} \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$. ▷ Level-zero encoding of a random coset
for $i = 0$ to k **do**
  Choose $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ ▷ Level-zero encoding of a random coset
  Choose $\mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma^*}$ ▷ Randomization vector
  Set $u_i = \left[\mathbf{e}_i\mathbf{y} + \sum_{j=1}^{m} r_j\mathbf{x}_j\right]_q$ ▷ Randomized lvl-1 encoding of $\mathbf{e}_i$
end for
Set $u^* = \left[\prod_{i=1}^{k} u_i\right]_q$ ▷ Level-k encoding of last k elements product
Set $v = [\mathbf{e}_0 \cdot u^*]$ ▷ Level-k encoding of the $k+1$ elements
Set $v' = [\mathbf{f} \cdot u^*]$ ▷ Level$-k$ encoding of other $k+1$ elements
Output: $(u_0, u_1, ..., u_k, v, v')$.

2. **Graded DDH (GDDH).** *The GDDH problem is hard for a Graded Encoding System GES if for any probabilistic polynomial running time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the distributions:*

$$(\text{params}, \mathbf{p}_{zt}, u_0, ..., u_k, v\ ) \text{ and}$$
$$(\text{params}, \mathbf{p}_{zt}, u_0, ..., u_k, v')$$

*is negligible in λ, where $(params, \boldsymbol{p}_{zt}\ ) \leftarrow InstGen(1^\lambda, 1^k)$ and $(u_0, u_1, ..., u_k, v, v') =$ **genGDDH**$(params, \boldsymbol{p}_{zt})$.*

3.5 Applications

Chapter 4

Security

Conclusions

Bibliography

[1] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete gaussian left-over hash lemma over infinite domains. Cryptology ePrint Archive, Report 2012/714, 2012.

[2] Ciprian Băetu, Petru Cehan, and Dan Mărculeţ. *On Bilinear Groups of Composite Order*, pages 389–398. Military Technical Academy Publishing House, 2016.

[3] Ioan Băetu and Ciprian Băetu. *Capitole speciale de algebră*. Taida, 1 edition, 2015.

[4] Dan Boneh and Matt Franklin. *Identity-Based Encryption from the Weil Pairing*, pages 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

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

[6] Xavier Boyen. *Attribute-Based Functional Encryption on Lattices*, pages 122–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[7] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. *Cryptanalysis of the Multilinear Map over the Integers*, pages 3–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[8] Jung Hee Cheon and Dong Hoon Lee. A note on self-bilinear maps. *Bulletin of the Korean Mathematical Society*, 46(2):303–309, 3 2009.

[9] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. *Practical Multilinear Maps over the Integers*, pages 476–493. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[10] Sanjam Garg. *Candidate Multilinear Maps*. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA, 2015.

[11] Sanjam Garg, Craig Gentry, and Shai Halevi. *Candidate Multilinear Maps from Ideal Lattices*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[12] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. *Attribute-Based Encryption for Circuits from Multilinear Maps*, pages 479–499. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[13] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[14] Jeffrey Hoffstein, Jill Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[15] Ferucio Laurenţiu Ţiplea and Constantin Cătălin Drăgan. Key-policy attribute-based encryption for boolean circuits from bilinear maps. In *Cryptography and Information Security in the Balkans - First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers*, pages 175–193, 2014.

[16] Antoine Joux. *A One Round Protocol for Tripartite Diffie–Hellman*, pages 385–393. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[17] A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *MATH. ANN*, 261:515–534, 1982.

[18] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Cryptology ePrint Archive, Report 2012/230, 2012.

[19] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 372–381, Washington, DC, USA, 2004. IEEE Computer Society.

[20] Ron Rothblum. On the circular security of bit-encryption. Cryptology ePrint Archive, Report 2012/102, 2012.

[21] Fré Vercauteren. Final report on main computational assumptions in cryptography.