NTNU – Trondheim
Norwegian University of
Science and Technology

On Fully Homomorphic Encryption

Prastudy Fauzi

Master in Security and Mobile Computing
Submission date: June 2012
Supervisor: Danilo Gligoroski, ITEM
Co-supervisor: Helger Lipmaa, University of Tartu

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Name of student: Prastudy Mungkas Fauzi

The student will analyse the most recent developments in fully homomorphic encryption. The student will start with the basic mathematical concepts used in the latest fully homomorphic cryptosystems based on learning with errors, and the main ideas common to these cryptosystems. Later, the student will make a comparison of the latest schemes and discuss some possible improvements. The student will conclude with some possible applications of fully homomorphic encryption.

Assignment given : 15 January 2012

Supervisor : Dr. Helger Lipmaa, Prof. Danilo Gligoroski

Abstract

In this thesis, we do a survey of the most recent fully homomorphic encryption schemes. We study some of the latest fully homomorphic encryption schemes, make an analysis of them and make a comparison. We start with Gentry's scheme, which was the first fully homomorphic encryption scheme, and choose four other fully homomorphic encryption schemes to analyze. We discuss the main ideas of each scheme, and how each scheme improves upon the previous ones. Whenever possible, we rewrite the main results of these schemes in a more detailed and readable format.

Contents

Acknowledgements

I would like to thank my main supervisor Helger Lipmaa for the time and effort he has put into guiding me in my work. He has continually given me valuable feedback on which recent work to take a closer look upon, which details I am missing from my work, how to write correctly in LaTeX and, along with Bingsheng Zhang, engaged me in informative discussions on lattice-based cryptography in general, and fully homomorphic encryption in particular. I would also like to thank my second supervisor Danilo Gligoroski for his feedback in writing this master's thesis.

Chapter 1

Introduction

1.1 Problem Statement

Cryptography, generally speaking, is the study of secret writing. As such, one important aspect of cryptography is encryption, which is the process of converting readable information into something unreadable. The readable information is known as plaintext, the unreadable output is known as ciphertext, and the conversion is done using a so called encryption algorithm. Encryption requires additional information to perform, which is known as the encryption key. For encryption to be useful, there must be a way to convert the ciphertext back to the plaintext. This process is called decryption, and it usually requires some additional knowledge that only privileged parties have access to, which is called the decryption key.

The decryption key used in a decryption algorithm may or may not be the same as the encryption key. When the decryption key is the same as the encryption key, we have symmetric encryption. When these two keys are different, we have public-key encryption. There are then two types of cryptography based on the type of encryption used: symmetric cryptography and public-key cryptography.

Symmetric cryptography is typified by the use of a common key between the sender and receiver. The most used encryption schemes today in symmetric cryptography are known to be very efficient, but difficult to break in a short amount of time. However, symmetric cryptography has some drawbacks. The main challenge is distributing the common key: the sender and receiver must somehow ensure that they share the same key, and that in the process of sharing this key no other party knows anything about it. This will be true for every pair of parties who want to communicate, meaning every party will also have to store many common keys. For security reasons, a common key is used for only a short amount of time, usually called one session.

Public-key cryptography, meanwhile, uses two different keys: a public key and the secret key. A public key is published by a party who wants to receive a message, and has the corresponding secret key that no one else knows. The public key and secret key are mathematically related in such a way that even if a party knows another party's public key, finding out the corresponding secret key is extremely difficult. The main advantage of public key cryptography is that key distribution is very easy. A party only

has to publish his public key to a common server that all parties trust, and anyone who wants to communicate simply uses this public key to send messages. However, public key encryption schemes are much less efficient than symmetric encryption schemes.

Encryption started out as a tool used mainly for military purposes, that is to send secret messages containing military information. Over time, it has been used much more widely. We use encryption very frequently in everyday life, from doing transactions with our favourite bank, to communicating privately with other individuals using email or instant messaging. One interesting use of encryption is in cloud computing. At a glance, it seems to be a very convenient way to store data and use cloud services to make use of the data. However, current implementations of cloud computing require a user to trust the cloud provider, who can get access to a user's private data if required. Storing the data in an encrypted form does not help, as the cloud service will not be able to do most of its operations on encrypted data without decrypting it first. If it were possible to store the data in an encrypted form while still enabling the cloud services to do operations on it, this trust requirement could be removed. Fully homomorphic encryption is a way of solving this challenge.

Fully homomorphic encryption is an encryption scheme where a party can receive encrypted data and perform arbitrary operations on this data efficiently. The data remains encrypted throughout, but the operations can be done regardless, without having to know the decryption key. Such a scheme would be very advantageous, for example in ensuring the privacy of data that is sent to a third-party service. This is in contrast with schemes like Paillier [Pai02] where you can not perform a multiplication of encrypted data without decrypting the data first, or ElGamal [Gam84] where you can not perform an addition of encrypted data without decrypting the data first.

Fully homomorphic encryption is a very new area of research: the first such scheme was constructed by Gentry [Gen09] in 2009. Gentry's scheme used ideals over polynomial rings, with security related to that of ideal lattices. Gentry's idea consists of two parts. First, define addition and multiplication on the ciphertext, in this case ordinary addition and multiplication over a polynomial ring. This will create a somewhat homomorphic scheme, which can evaluate circuits of additions and multiplications up to a certain depth. However, every multiplication operation increases the noise by a significant amount, meaning that at some point the noise will be too big. The second idea tries to solve this by doing noise reduction. Gentry modified his scheme to be bootstrappable, that is it can evaluate its own decryption circuit. He then showed that any somewhat homomorphic scheme that is bootstrappable can be changed into a fully homomorphic scheme, as by the use of bootstrapping, the noise in any ciphertext can be reduced to be the same noise of evaluating the decryption circuit.

Since Gentry's breakthrough, there have been many advances inspired by Gentry's work. The latest fully homomorphic encryption schemes use public key cryptography and are based on lattices. Lattice-based cryptography is gaining more interest due to its security against quantum computers, and its worst case security guarantee. However, the main problem remains: the schemes do not yet have an efficient implementation that still maintains adequate security requirements. Seen in this light, recent advances in fully homomorphic encryption either improves the efficiency of previous schemes, or proposes

a new scheme with better efficiency.

1.2 Outline

The work in this thesis consists of three parts. First, we will look at the preliminary knowledge required to understand the later sections on fully homomorphic encryption. We start with definitions on algebraic number theory, lattice theory, and public key encryption. We then move to an important security assumption named learning with errors. Learning with errors is a lattice-based security assumption introduced in [Reg05], and it can be shown to be related to the hardness of the shortest vector problem in lattice theory.

Second, we will discuss in detail the most recent fully homomorphic encryption schemes. The main elements that will be analyzed here are the use of learning with errors and its variant ring-learning with errors, and techniques such as key switching and modulus reduction. We will also show how each scheme improves upon the previous schemes that we have discussed.

One work we start with is the scheme of Brakerski and Vaikuntanathan [BV11], which improves upon Gentry's scheme by the introduction of relinearization and modulus switching, removing the need of a squashing step. We continue with the scheme of Brakerski, Gentry, and Vaikuntanathan [BGV12], which improves upon Brakerski and Vaikuntanathan's scheme by having a general scheme that can be used both in the learning with errors setting, or its ring variant, and by improving the relinearization and modulus switching techniques to obtain fully homomorphic encryption without bootstrapping. We will then continue with Brakerski's scheme [Bra12], which uses Regev's LWE-based scheme under the invariant perspective, and modulus switching is not required. Finally, we focus on Fan and Vercauteren's scheme [FV12], that implements Brakerski's scheme in the ring learning with errors setting, improving its efficiency.

Apart from Gentry's scheme, the schemes that we choose to discuss are very new. Brakerski and Vaikuntanathan's scheme [BV11] was published in October 2011, Brakerski, Gentry, and Vaikuntanathan's scheme [BGV12] was published in January 2012, while both Brakerski's work [Bra12] and Fan and Vercauteren's work [FV12] are still only available as eprints.

Third, we will analyze and compare the bounds, size and complexity of the chosen schemes. A discussion of possible improvements for future work, and a look of possible applications of fully homomorphic encryption will also be included.

1.3 Author's Contribution

This work acts as a survey of the most recent fully homomorphic encryption schemes. We study some of the latest fully homomorphic encryption schemes, make an analysis of them and make a comparison. Whenever possible, we rewrite the main results of these schemes in a more detailed and readable format. This includes the proof of Lemma 3.4.1, Lemma 3.4.2, Lemma 3.5.1, Lemma 3.5.2, and Lemma 3.6.1, the observations of

the performance for each of the schemes, and the comparison of the schemes. We hope this work can help readers be up to date with the field of fully homomorphic encryption, paving way to further advances in the field.

For the author, this work will serve as a foundation for ongoing and future work. One direction we are working on is finding a tighter bound for the expansion factor than those used in [BGV12]. This will be discussed in Section 4.1. Another way to proceed is to go deeper into the idea of batching, which will be discussed in Section 4.2. A third way is to improve the bounds given in [FV12], which is the most recent result we focus on in this work.

Chapter 2

Preliminaries

Before we go into the encryption schemes, we will give an overview of the mathematical concepts used in the later sections. This will mostly consist of definitions in algebraic number theory and lattice theory, then continue with learning with errors and its ring variant.

2.1 Basic Definitions

2.1.1 Elementary Number Theory.

Let n, x be positive integers. The division algorithm states that there are unique integers a, b such that

$$x = an + b, \text{with} -\frac{n}{2} < b \leq \frac{n}{2}.$$

If $b = 0$, we say that n *divides* x, and that n is a *divisor* of x. If x has no divisors except 1 and x itself, we say that x is *prime*.

Define $[x]_n = b$, where $b \in \mathbb{Z}_n$ satisfies the above requirement. Moreover, if we have values $c = (c_1, \cdots, c_k)$ with $c_1, \ldots, c_k \in \mathbb{Z}$, define

$$[c]_q = ([c_1]_q, \ldots, [c_k]_q) = (c'_1, \ldots, c'_k), \text{with} -\frac{q}{2} < c'_i \leq \frac{q}{2}.$$

Let a be a real number. Then we define three operations below:

- Define the floor function $\lfloor a \rfloor$ to be the largest integer which is not larger than a. This is also known as the integer part of a.
- Define the ceiling function $\lceil a \rceil$ to be the smallest integer which is not smaller than a.
- Define the round function $\lfloor a \rceil$ to be the closest integer to a.

For example, $\lfloor 1.49 \rfloor = \lfloor 1.49 \rceil = 1$, and $\lceil 1.51 \rceil = \lfloor 1.51 \rceil = 2$.

Given two integers a, b not both zero, we define the two operations below:

- Define the *greatest common divisor* of a and b, denoted $gcd(a, b)$, to be the largest positive integer that divides both a and b.

- Define the *lowest common multiple* of a and b, denoted $lcm(a, b)$, to be the smallest positive integer that is both a multiple of a and a multiple of b.

For example, $gcd(9, 12) = 3$, $gcd(2, 7) = 1$, $lcm(9, 12) = 36$, and $lcm(2, 7) = 14$.

When $gcd(a, b) = 1$, we say that a and b are *relatively prime*. For example, 2 and 7 are relatively prime, but 9 and 12 are not relatively prime.

For a positive integer n, define *Euler's totient function* $\phi(n)$ to be the number of positive integers not greater than n but relatively prime to n. For example, 6 is relatively prime to 1 and 5, but not relatively prime to $2, 3, 4$, or 6, so $\phi(6) = 2$.

We will present two theorems related to the totient function, which will be used in later sections. The first one is useful to compute $\phi(n)$, while the second one is useful to simplify modular exponentiation.

Theorem 2.1.1 *Let n be a positive integer which can be written as $n = p_1^{a_1} \cdot p_2^{a_2} \cdots p_k^{a_k}$, where $p_1, p_2, \ldots, p_k$ are the k distinct prime factors of n. Then we have*

$$\phi(n) = n \cdot (1 - \frac{1}{p_1}) \cdot (1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}).$$

Theorem 2.1.2 *(Euler's Theorem) Let n, a be positive integers which are relatively prime. Then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

For a positive integer n, define the *Carmichael function* $\lambda(n)$ to be the smallest positive integer m such that

$$a^m \equiv 1 \pmod{n}.$$

for an integer a relatively prime to n.

Carmichael's theorem is a way to easily compute $\lambda(n)$.

Theorem 2.1.3 *(Carmichael's Theorem) Let n be a positive integer. Then*

$$\begin{aligned} \lambda(n) &= \phi(n), n = 2, 4, p^k, 2p^k, p \text{ odd prime}, \\ \lambda(n) &= \frac{1}{2}\phi(n), n = 2^k, k \geq 3, \text{and} \\ \lambda(lcm(a, b)) &= lcm(\lambda(a), \lambda(b)). \end{aligned}$$

2.1.2 Algebraic Structures.

A *group* $(G, +)$ is an algebraic structure where:

1. The operation $(+)$ is closed and associative in G,
2. There exists an identity element $0 \in G$ and inverse element $-a$ for each element $a \in G$.

If the operation $(+)$ is also commutative, we say that $(G, +)$ is an *abelian group*. For an integer n and an element $g \in G$, define ng to be the result of:

- $g + g + \cdots + g$ (n times), when $n > 0$,
- $(-g) + (-g) + \cdots + (-g)$ ($-n$ times), when $n < 0$, or
- 0, when $n = 0$.

If every element $a \in G$ can be written as $a = ng$ for some $n \in \mathbb{Z}$, we say that $(G, +)$ is a *cyclic group*. In this case, g is said to be a *generator* of the group.
A *ring* $(R, +, \cdot)$ is an algebraic structure that satisfies the following conditions:

1. $(R, +)$ is an abelian group.
2. $(R, \cdot)$ is associative.
3. The distributive laws apply to $(R, +, \cdot)$.

We usually work with rings which are commutative and have an identity element under the operation $(\cdot)$.

Given a ring $(R, +, \cdot)$, a subset I of R is called an *ideal* if it satisfies the following conditions:

1. $(I, +)$ is a subgroup of $(R, +)$.
2. For any two elements $x \in I, r \in R$, $x \cdot r \in I$ and $r \cdot x \in I$.

For example, in the ring $R = \mathbb{Z}$, the ideal $I = 2\mathbb{Z}$ is the set of even integers.

Given $a \in R$ and an ideal I, we can define the *equivalence class*

$$[a] = \{a + x | x \in I\}.$$

Then $[a] = [b] \iff a - b \in I$. The set of all distinct equivalence classes is the *quotient ring* R/I. For example, in the ring $R = \mathbb{Z}$, with ideal $I = 2\mathbb{Z}$, the quotient ring $R/I = \mathbb{Z}/2\mathbb{Z}$ has two equivalence classes $[0] = \{\ldots, -4, -2, 0, 2, 4, \ldots\}$ and $[1] = \{\ldots, -3, -1, 1, 3 \ldots\}$.

A *field* $(F, +, \cdot)$ is a commutative ring which under $(\cdot)$ has an identity element and inverses. $(F, +, \cdot)$ is a field iff $(F, +)$ and $(F - \{0\}, \cdot)$ are both abelian groups and the distributive laws apply. We will mostly use the fields $\mathbb{Z}_q$ (where q is a prime number) and GF(2) (the Galois field of two elements), which is mostly used for studying arithmetic circuits with operations XOR and AND. Here, XOR is associated with the addition operator, while AND is associated with the multiplication operator.

A *polynomial ring* $F[X]$ is a ring formed from a set of polynomials in the variable X, where the coefficients are from a field F. If $f(X) = a_d X^d + \cdots + a_1 X + a_0 \in F[X]$ is an irreducible polynomial, we have the quotient ring $R = F[X]/(f(X))$. Moreover, when F is the field $\mathbb{Z}_q$ we write $R_q = \mathbb{Z}_q[X]/(f(X))$. Additionally, if we have that f has degree d, then $|R_q| = q^d$.

2.1.3 Inner Products.

Let V be an n-dimensional vector space over a field F. For $\boldsymbol{a} = (a_1, \ldots, a_n)^T, \boldsymbol{b} = (b_1, \ldots, b_n)^T \in V$, define the inner product

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_i b_i.$$

We will mostly use the polynomial ring $R[x] = \mathbb{Z}_q[x]/(f(x))$, where f is a monic polynomial (polynomial with leading coefficient 1) with degree d. In this case,

$$\boldsymbol{a} = \sum_{i=0}^{d-1} a_i x^i, \boldsymbol{b} = \sum_{i=0}^{d-1} b_i x^i.$$

2.1.4 Norms.

Let $s = \sum_{i=0}^{d} s_i x^i$ be an element of a polynomial ring R. Define the *Euclidean norm*

$$\|s\|_2 = \sqrt{\sum_{i=0}^{d} s_i^2},$$

and the *infinity norm*

$$\|s\|_\infty = \max_i |s_i|.$$

Also, for $\boldsymbol{x} \in R^n$, we define the ℓ_1-norm

$$\ell_1(\boldsymbol{x}) = \sum_{i=1}^{n} \|x_i\|_2.$$

We will be using norms in many of the inequalities in this work. For that reason, we will give two well-known inequalities related to norms that we will be using often in the following sections.

Theorem 2.1.4 *(Triangle Inequality) Given two vectors $\boldsymbol{a}, \boldsymbol{b}$ of the same size, we have that*

$$\|a + b\| \leq \|a\| + \|b\|.$$

Theorem 2.1.5 *(Cauchy-Schwarz) Given two non-zero vectors $\boldsymbol{a}, \boldsymbol{b}$ of the same size, we have that*

$$|\langle a, b \rangle| \leq \|a\| \|b\|,$$

where equality holds if and only if $\boldsymbol{a} = k\boldsymbol{b}$ for some scalar k.

2.1.5 Expansion Factor.

Let R be a polynomial ring. The expansion factor of R is defined as

$$\gamma_R = \max\{\frac{\|a \cdot b\|_2}{\|a\|_2 \|b\|_2} : a, b \in R\}.$$

When $R = \mathbb{Z}[x]/(x^d + 1)$, we can prove using Cauchy-Schwarz that $\gamma_R \leq \sqrt{d}$.

2.1.6 Negligible Functions.

A function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive integer c there exists an integer $N = N(c)$ such that for all $x > N$,

$$|f(x)| \leq \frac{1}{x^c}.$$

We usually denote negligible functions as $negl(x)$.

2.2 Definitions from Lattice Theory

Lattice theory is a study of mathematical structures called lattices. They have many interesting applications in cryptography, and as we will see, some lattice problems have convenient properties in terms of complexity.

2.2.1 Lattice.

A *lattice* is a set of points in n-dimensional space with a periodic structure. As such, it is a discrete subgroup of $\mathbb{R}^n$ under addition of vectors in $\mathbb{R}^n$.

Let $b_1, b_2, \cdots, b_k$ be k linearly independent vectors in $\mathbb{R}^n$. Then we can define the lattice generated by these vectors as

$$L(b_1, b_2, \cdots, b_k) = \{\sum a_i b_i | a_i \in \mathbb{Z}\}$$

By this definition, $\{b_1, b_2, \cdots, b_k\}$ form a basis of this lattice, which has dimension k. Every lattice has a basis, but this basis is not unique. For example, if $\{b_1, b_2\}$ is a basis of a lattice L in $\mathbb{R}^2$ then $\{b_1, b_1 + b_2\}$ is also a basis of L. In general, if B is a basis of a lattice L of dimension n, and $U_{n\times n}$ is an integer matrix of determinant 1, then BU is also a basis of L.

A cryptographic construction using lattices can have strong provable security guarantees based on the *worst-case hardness* of lattice problems. This is done by having parameters chosen such that breaking the construction is as hard as solving lattice problems in the worst case [MR08]. One of the most efficient ones are cryptosystems based on learning with errors, which will be discussed later.

2.2.2 Shortest Vector Problem.

One important property of a lattice is the length of the shortest non-zero vector $v \in L$, denoted as $\lambda_1(L)$. Here we use the Euclidean norm $\|\cdot\|_2$

This leads to the shortest vector problem (SVP): Given a lattice L with basis $B = (b_1, b_2, \cdots, b_k)$, find a vector $v \in L$ such that $v = \lambda_1(L)$.

The hardness of SVP depends on the basis used. One algorithm that approximate a solution for SVP is the LLL algorithm. The strength of LLL is that it runs in polynomial time, but with a good choice of a basis, the LLL algorithm may reach errors of up to an exponential factor $2^{O(n)}$, where n is the dimension of the lattice [MR08].

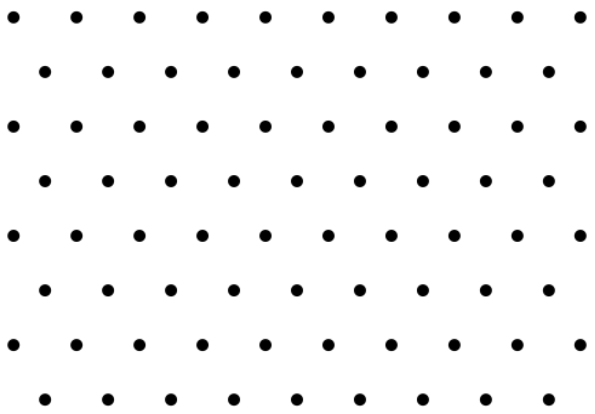

Figure 2.1: 2-dimensional lattice with base $\{(1, 0), (\frac{1}{2}, \frac{1}{2}\sqrt{3})\}$

2.2.3 GapSVP.

Another problem related to SVP is to determine whether the length of the shortest non-zero vector in L is at most 1 or larger than $\beta > 1$. This is known as $GapSVP_\beta$: Given a basis $B = (b_1, b_2, \cdots, b_k)$ of a lattice L, decide whether $\lambda_1(L) \leq 1$ or $\lambda_1(L) > \beta$.

SVP and GapSVP are NP-hard problems, and no efficient quantum algorithm has yet been found that solves these problems or approximates them with a small error [MR08].

2.3 Learning With Errors

Suppose we want to get values of the form $(\boldsymbol{a}, b) \in \mathbb{Z}_q^{n+1}$, where $\boldsymbol{a} \in \mathbb{Z}_q^n, b \in \mathbb{Z}_q$. Consider the two distributions below:

1. Select random values $(\boldsymbol{a_i}, b_i)$ uniformly from $\mathbb{Z}_q^{n+1}$.
2. Uniformly choose $s \in \mathbb{Z}_q^n$. Select random a_i uniformly from $\mathbb{Z}_q^n$ and $e_i \in \mathbb{Z}_q$ from some distribution χ over $\mathbb{Z}_q$. Set $b_i = \langle \boldsymbol{a_i}, s \rangle + e_i \in \mathbb{Z}_q$. Give the values $(\boldsymbol{a_i}, b_i)$.

The learning with errors assumption $\text{LWE}_{n,q,\chi}$ states that given samples from the second distribution, we cannot approximate the value of s. A variant of this problem, the decision learning with errors assumption $\text{DLWE}_{n,q,\chi}$ states that these two distributions are indistinguishable. Regev [Reg05] proved that by choosing correct parameters n, q, χ, $\text{LWE}_{n,q,\chi}$ is as hard as the shortest vector problem, and that LWE and DLWE are equivalent provided that the prime q is bounded by a polynomial in n.

2.4 Ring Learning With Errors

We will use the variation of ring learning with errors (RLWE) used in [BGV12]. Let λ be the security parameter, and $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer satisfying $q \equiv 1 \mod d$. Let $R = \mathbb{Z}[x]/(f(x))$ and let $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be some distribution over R.
Consider the two distributions below:

1. Sample values (a_i, b_i) uniformly from R_q^2.
2. Uniformly choose $s \in R_q$. Sample a_i uniformly from R_q and $e_i \in R$ from χ. Set $b_i = a_i \cdot s + e_i \in R_q$. Give the values (a_i, b_i).

The ring learning with errors assumption $\text{RLWE}_{d,q,\chi}$ states that these two distributions are indistinguishable. The importance of the RLWE comes from the fact that by choosing $B = \omega(\sqrt{d \log d})$ and χ that outputs elements of R with length at most B (except for a negligible probability), the worst case shortest vector problem over ideal lattices, which are lattices corresponding to ideals I of polynomial ring R, can be reduced to RLWE [LPR10].

LWE is the more standard assumption and is a harder problem than RLWE, but RLWE can be shown to be more efficient [LPR10]. Notice that LWE uses elements in $\mathbb{Z}_q^{n+1}$, but RLWE only uses elements in R_q^2. This means that RLWE uses smaller keys, and arithmetic in R_q is more efficient.

2.5 Public Key Cryptosystem

A public key cryptosystem consists of three elements: key generation, encryption, and decryption.

- Key generation is the process of generating a public key and secret key pair for encryption and decryption. It requires a security parameter λ, typically the size of the resulting public key.
- Encryption is the function that maps a plaintext into a ciphertext, using a public key. The domain of this encryption function is called plaintext space.
- Decryption is the function that maps a ciphertext back into plaintext, using a secret key.

A public key cryptosystem with a key generation algorithm KG, encryption algorithm E and decryption algorithm D can then be written as (KG, E, D).

2.6 Probabilistic Encryption

A *probabilistic encryption scheme* is an encryption scheme which introduces randomness in the encryption algorithm. This is done so that encrypting the same message more than once will result in different ciphertexts, making it difficult to detect two different encryptions of the same message. The randomness factor introduced in the encryption of a particular message is often called the *noise*. We define $E_{pk}(m; r)$ to be an encryption of a message m using the encryption algorithm E and public key pk, with noise parameter r.

For a probabilistic public key cryptosystem (KG, E, D) and adversary A, and let λ be the security parameter. Consider the following two games:

1. Game 1:
   - Set $(pk, sk) \leftarrow KG(1^\lambda)$.
   - Get $(m_1, m_2) \leftarrow A(pk)$.
   - Output $E_{pk}(m_1; r)$ for random noise r.
2. Game 2:
   - Set $(pk, sk) \leftarrow KG(1^\lambda)$.
   - Get $(m_1, m_2) \leftarrow A(pk)$.
   - Output $E_{pk}(m_2; r)$ for random noise r.

(KG, E, D) is *indistinguishable under chosen plaintext attack* (IND-CPA) if for all adversaries A running in polynomial time, there is no way to distinguish between these two games except with negligible probability. That is, for all adversaries A and security parameter λ, there exists a negligible function f such that:

$$|Pr[A = 1 : \text{Game 1}] - Pr[A = 1 : \text{Game 2}]| \leq f(\lambda).$$

2.7 Homomorphic Encryption

We will give the definition of homomorphic encryption under addition and multiplication, and show two schemes that do not quite match the criteria: homomorphic under one operation but not the other.

2.7.1 Definition

Let the plaintext space P have "addition" operator $+$, and "multiplication" operator $\times$, and let the ciphertext space C have "addition" operator $\oplus$, and "multiplication" operator $\otimes$. Let $E : P \to C$ be a probabilistic encryption scheme, and $D : C \to P$ the corresponding decryption scheme.
A public key cryptosystem (KG, E, D) is *homomorphic* under addition and multiplication, if

$$D(E(a) \oplus E(b)) = a + b$$

and

$$D(E(a) \otimes E(b)) = a \times b$$

for all $a, b \in P$.

In general, an encryption scheme homomorphic under addition and multiplication has a homomorphic evaluation function $f : C^n \to C$ that when decrypted will result in a corresponding function $g : P^n \to P$ where $D(f(c_1, \cdots, c_n)) = g(p_1, \cdots, p_n)$ with $c_i = E(p_i)$, and g is f with $\oplus$ replaced by $+$, $\otimes$ replaced by $\times$.

Typically, we use a public key encryption scheme with public key pk and secret key sk, and the evaluation function f might also need an evaluation key evk. These keys are generated by a key generator $(pk, sk, evk) \leftarrow KG(1^\lambda)$, where λ is the security parameter. In this case, the encryption function is $E_{pk} : P \to C$.

2.7.2 Homomorphic Under Addition: Paillier

Paillier is a public key cryptosystem that relies on the Decisional Composite Residuosity Assumption (DCRA) [Pai02]: given a composite integer n and integer $x \in \mathbb{Z}_{n^2}$, it is hard to decide whether or not there is a $y \in \mathbb{Z}_{n^2}$ such that

$$x \equiv y^n \pmod{n^2}.$$

Key generation in Paillier is as follows:

1. Generate distinct prime numbers of the same size p, q, and let $n = p \cdot q$. Ensure that n is an integer where DCRA holds.
2. Set $\lambda = lcm(p-1, q-1)$. The public key is n.
3. For the corresponding secret key, compute, $\mu = \lambda^{-1} \mod n$ The secret key is (λ, μ).

To encrypt a message $m \in \mathbb{Z}_n$, we do the following:

1. Select a random $r \in \mathbb{Z}_n^*$.
2. Output the ciphertext $E(m; r) = (n+1)^m r^n \mod n^2$.

Under the DCRA assumption, this scheme is IND-CPA secure.

To decrypt a ciphertext $c \in \mathbb{Z}_{n^2}^*$, simply compute $m = L(c^\lambda \mod n^2) \cdot \mu$, where L is the discrete logarithm function. In Paillier, this function can be simplified as $L(u) = \lfloor \frac{u-1}{n} \rfloor$. Decryption works, because due to Carmichael's Theorem,

$$\begin{aligned} \lambda(n^2) &= lcm(\lambda(p^2), \lambda(q^2)) \\ &= lcm(p^2 - p, q^2 - q) \\ &= pq \cdot lcm(p-1, q-1) \\ &= n \cdot \lambda. \end{aligned}$$

So by definition $r^{n\cdot\lambda} \equiv 1 \pmod{n^2}$. So we have

$$\begin{aligned} L(c^\lambda \mod n^2) \cdot \mu &= L((n+1)^{m\cdot\lambda} \cdot r^{n\cdot\lambda} \mod n^2) \cdot \mu \\ &= L((n+1)^{m\cdot\lambda} \mod n^2) \cdot \mu \\ &= m \cdot \lambda \cdot \mu \\ &\equiv m \pmod{n}. \end{aligned}$$

Note that if we have two ciphertexts $E(m_0; r_0)$ and $E(m_1; r_1)$ which are encryptions of m_0, m_1 respectively, then

$$\begin{aligned} E(m_0; r_0) \cdot E(m_1; r_1) &\equiv ((n+1)^{m_0} {r_0}^n) \cdot ((n+1)^{m_1} {r_1}^n) \\ &\equiv (n+1)^{m_0+m_1} (r_0 \cdot r_1)^n \\ &\equiv E(m_0 + m_1; r_0 \cdot r_1) \pmod{n^2}. \end{aligned}$$

Then $D(E(m_0; r_0) \cdot E(m_1; r_1)) = m_0 + m_1$. So Paillier is an additively homomorphic scheme: given encryptions of m_0 and m_1, we can get an encryption of $m_0 + m_1$ without having to know the secret key. However, given encryptions of m_0 and m_1 there is no known way of obtaining an encryption of $m_0 \cdot m_1$ without knowing m_0 or m_1 first. So Paillier is not known to be homomorphic under multiplication.

2.7.3 Homomorphic Under Multiplication: ElGamal

ElGamal [Gam84] is a public key cryptosystem that relies on the hardness of the decisional Diffie-Hellman (DDH) problem [Bon98]: Let G be a group with generator g. Then it is hard to distinguish between the distributions $\langle g^a, g^b, g^{ab} \rangle$ and $\langle g^a, g^b, g^c \rangle$ where a, b, c are integers chosen randomly from $[1, |G|]$

Key generation in ElGamal is as follows:

1. Let $\mathbb{G}$ be a cyclic group with prime order q, where the DDH assumption holds.
2. Let g be a generator of $\mathbb{G}$ chosen randomly from $\mathbb{Z}_q^*$.
3. Generate the secret key $s \leftarrow \mathbb{Z}_q$, and the public key $h = g^s$.

To encrypt a message $m \in \mathbb{G}$, we do the following:

1. Select a random $r \leftarrow \mathbb{Z}_q$.
2. Compute $c_1 = g^r$.
3. Compute $c_2 = m \cdot h^r$.
4. Output $E(m; r) = (c_1, c_2)$.

Under the DDH assumption, this scheme is IND-CPA secure [Bon98].

To decrypt a ciphertext $c = (c_1, c_2)$, compute $m = D(c) = c_2 \cdot (c_1^s)^{-1}$. Decryption works, because

$$\begin{aligned} c_2 \cdot (c_1^s)^{-1} &= (m \cdot h^r) \cdot (g^r)^{-s} \\ &= (m \cdot h^r) \cdot (g^s)^{-r} \\ &= (m \cdot h^r) \cdot h^{-r} \\ &= m. \end{aligned}$$

Note that if we have two ciphertexts $E(m_0; r_0)$ and $E(m_1; r_1)$, then

$$\begin{aligned} E(m_0; r_0) \cdot E(m_1; r_1) &= (g^{r_0} \cdot g^{r_1}, (m_0 \cdot h^{r_0}) \cdot (m_1 \cdot h^{r_1})) \\ &= (g^{r_0 + r_1}, (m_0 \cdot m_1) \cdot h^{r_0 + r_1}) \\ &= E(m_0 \cdot m_1; r_1 + r_2). \end{aligned}$$

Then $D(E(m_0; r_0) \cdot E(m_1; r_1)) = m_0 \cdot m_1$. So ElGamal is a multiplicatively homomorphic scheme: given encryptions of m_0 and m_1, we can get an encryption of $m_0 \cdot m_1$

without having to know the secret key. However, given encryptions of m_0 and m_1 there is no known way of obtaining an encryption of $m_0 + m_1$ without knowing both m_0 and m_1 first. So ElGamal is not known to be homomorphic under addition.

Chapter 3

Recent Developments

This section will discuss the latest fully homomorphic encryption schemes, and a comparison of complexities for these schemes. We will start with Gentry's scheme and his bootstrapping theorem, how bootstrapping could reduce noise and how this theorem was a blueprint for later results, until schemes were found that did not require bootstrapping.

3.1 Fully Homomorphic Encryption

We will use the definitions in [BV11]. A scheme is *somewhat homomorphic* if it can achieve homomorphism under addition and multiplication, without doing any noise reduction, i.e., without any process of reducing the size of the noise relative to the size of the ciphertext.

An encryption scheme is *compact* if there exists a polynomial (over the security parameter λ) $p = p(\lambda)$ such that the output of the evaluation function is at most p bits long, regardless of f or the number of inputs.

Moreover, an encryption scheme is *fully homomorphic* if it is compact and homomorphic for all arithmetic circuits over GF(2).

3.2 Gentry's Scheme [Gen09]

Gentry's scheme used an ideal I of a ring R, where the noise e is chosen to be an element in I, so that it has the form $e = rI$ for some $r \in R$. This means that the message m is encrypted to $m + rI$, and decrypting is the process of getting rid of the ideal. The homomorphic properties can be seen from the fact that if $c_1 = m_1 + r_1 I$ and $c_2 = m_2 + r_2 I$, then

$$\begin{aligned} c_1 + c_2 &= (m_1 + m_2) + (r_1 + r_2)I \\ c_1 c_2 &= (m_1 + r_1 I)(m_2 + r_2 I) = (m_1 m_2) + (m_1 r_2 + m_2 r_1 + r_1 r_2 I)I. \end{aligned}$$

Notice that after addition, the noise is $(r_1 + r_2)I$, while after multiplication the noise is dominated by $r_1 r_2 I$. This means that addition approximately doubles the noise, while multiplication approximately squares the noise. After a number of operations, the noise will overwhelm the ciphertext and make decryption incorrect. Gentry solved this problem

by evaluating the decryption function homomorphically with the ciphertext as input, which will create an equivalent ciphertext which has a small noise again. This is known as bootstrapping, and requires that the decryption function to be efficient, with the decryption circuit as simple as possible.

Gentry's scheme relied on the hardness assumptions on ideal lattices. The main drawback here is that the field of ideal lattices has not been very well studied. Also, there is a need for a squashing step to reduce the decryption complexity, but requires an additional strong assumption, that is the sparse subset-sum assumption: given a big set of integers S, a modulus M and a target sum t, it is difficult to find a sparse subset of S that sums up to $t \pmod M$. However, his work is very significant because it was the first scheme proved to be fully homomorphic, and because of his bootstrapping theorem [Vai11].

A homomorphic encryption scheme E is *bootstrappable* if it can evaluate its own decryption circuit, and slightly augmented versions of it. A PKE scheme is *weakly circular secure* if it is IND-CPA secure even for an adversary with additional information containing encryptions of all secret key bits $\{E(sk_i)\}$, where sk_i is the i-th bit of the secret key sk. Gentry's theorem states that if E is bootstrappable and the PKE is weakly circular secure, then E can be modified into a fully homomorphic encryption scheme.

This method of starting from a somewhat homomorphic encryption scheme and applying the bootstrapping theorem became a blueprint for many of the subsequent schemes.

3.3 Brakerski and Vaikuntanathan's Scheme [BV11]

3.3.1 Overview

The scheme uses relinearization to make it somewhat homomorphic. The noise is managed by modulus switching discussed below. The scheme is then shown to be bootstrappable which turns it into a fully homomorphic encryption scheme.

The most significant development of [BV11] compared to Gentry's scheme is the use of well-known security assumptions based on DLWE, and the introduction of the relinearization and modulus switching techniques. Modulus switching in particular removes the need of the expensive squashing step used in Gentry's scheme.

3.3.2 Encryption Scheme

Brakerski and Vaikuntanathan define an LWE-based public key encryption scheme as follows. Let λ be the security parameter, n be a positive integer polynomial in λ, k be a positive integer polynomial in n, and q an odd number sub-exponential in n. Let χ be a noise distribution that produces small numbers. We have the secret key $\boldsymbol{s} = (s[1], \cdots, s[n]) \in \mathbb{Z}_q^n$ and public key $(\boldsymbol{A}, \boldsymbol{v} = \boldsymbol{A}\boldsymbol{s} + 2\boldsymbol{e})$ where $\boldsymbol{A}$ is a $k \times n$ matrix chosen uniformly from $\mathbb{Z}_q^{k \times n}$ and $\boldsymbol{e}$ is chosen uniformly from χ^k.
j

Suppose $m \in \{0, 1\}$ is the bit we want to encrypt. To encrypt, we do the following:

1. Select a random $\boldsymbol{r} \in \{0,1\}^k$.
2. Compute $\boldsymbol{a} = \boldsymbol{A}^T\boldsymbol{r}$ and $b = \boldsymbol{v}^T\boldsymbol{r} + m$.
3. Output $(\boldsymbol{a}, b)$.

The ciphertext is an element in $\mathbb{Z}_q^{n+1}$ generated the same way as the distribution we have seen in Section 2.3. on learning with errors. Thus according to $\mathrm{DLWE}_{n,q,\chi}$ (where χ is a uniform distribution over $\mathbb{Z}_q$), we can use this scheme a polynomial number of times with negligible probability that an adversary can guess $\boldsymbol{s}$.

To decrypt a ciphertext $(\boldsymbol{a}, b)$, we do the following:

1. Compute $b' = b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = 2e + m \in \mathbb{Z}_q$ for some noise e.
2. Output $m = b' \mod 2$.

Decryption works because

$$\begin{aligned} b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle &= (\boldsymbol{v}^T\boldsymbol{r} + m) - (\boldsymbol{a}^T\boldsymbol{s}) \\ &= (\boldsymbol{v}^T\boldsymbol{r} + m) - (\boldsymbol{v}^T\boldsymbol{r} - 2\boldsymbol{e}^T\boldsymbol{r}) \\ &= 2\boldsymbol{e}^T\boldsymbol{r} + m \end{aligned}$$

so taking this value modulo 2 results in m.

3.3.3 Ideas

3.3.3.1 Relinearization (Key Switching).

Given a ciphertext $(\boldsymbol{a}, b), \boldsymbol{a} = (a[1], \cdots, a[n])$, consider the linear evaluation function $f_{\boldsymbol{a},b} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ as follows: $f_{\boldsymbol{a},b}(\boldsymbol{x}) = b - \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b - \sum_{i=1}^{n} \boldsymbol{a}[i] \cdot \boldsymbol{x}[i] \pmod q$, where the variables are $\boldsymbol{x} = (\boldsymbol{x}[1], \cdots, \boldsymbol{x}[n])$. Note that $m = f_{\boldsymbol{a},b}(\boldsymbol{s}) \pmod 2$. Now, one wants to make an evaluation function which is a combination of additions and multiplications of these f over different ciphertexts $(\boldsymbol{a_i}, b_i)$.
First note that

$$\begin{aligned} f_{(\boldsymbol{a},b)}(\boldsymbol{x}) + f_{(\boldsymbol{a'},b')}(\boldsymbol{x}) &= (b - \sum_{i=1}^{n} \boldsymbol{a}[i] \cdot \boldsymbol{x}[i]) + (b' - \sum_{i=1}^{n} \boldsymbol{a'}[i] \cdot \boldsymbol{x}[i]) \\ &= (b + b') - \sum_{i=1}^{n} (\boldsymbol{a}[i] + \boldsymbol{a'}[i]) \cdot \boldsymbol{x}[i] \\ &= f_{(\boldsymbol{a}+\boldsymbol{a'},b+b')}(\boldsymbol{x}), \end{aligned}$$

so addition in f is homomorphic.
However, multiplication seems problematic, as

$$\begin{aligned} f_{(\boldsymbol{a},b)}(\boldsymbol{x}) \cdot f_{(\boldsymbol{a'},b')}(\boldsymbol{x}) &= (b - \sum_{i=1}^{n} \boldsymbol{a}[i] \cdot \boldsymbol{x}[i]) \cdot (b' - \sum_{i=1}^{n} \boldsymbol{a'}[i] \cdot \boldsymbol{x}[i]) \\ &= h_0 + \sum_{i=1}^{n} h_i \cdot \boldsymbol{x}[i] + \sum_{1 \le i \le j \le n} h_{i,j} \cdot \boldsymbol{x}[i]\boldsymbol{x}[j], \end{aligned}$$

where

$$\begin{aligned} h_0 &= bb', \\ h_i &= -(b\boldsymbol{a'}[i] + b'\boldsymbol{a}[i]), \\ h_{i,j} &= \boldsymbol{a}[i]\boldsymbol{a'}[j] + \boldsymbol{a}[j]\boldsymbol{a'}[i]. \end{aligned}$$

The problem here is that the number of coefficients is $1 + n + \binom{n+1}{2} = \frac{(n+1)(n+2)}{2}$, so the ciphertext becomes quadratic in the size of s.
This can be overcome by a method called *relinearization* [BV11]. Suppose that the secret key $\boldsymbol{s} = (\boldsymbol{s}[1], \ldots, \boldsymbol{s}[n])$ is changed into a secret key

$$\boldsymbol{t} = (\boldsymbol{s}[1], \ldots, \boldsymbol{s}[n], \boldsymbol{s}[1]\boldsymbol{s}[1], \boldsymbol{s}[1]\boldsymbol{s}[2], \ldots, \boldsymbol{s}[n]\boldsymbol{s}[n]).$$

Then $h_{\boldsymbol{a},b}(\boldsymbol{x}) = f_{(\boldsymbol{a},b)}(\boldsymbol{x}) \cdot f_{(\boldsymbol{a'},b')}(\boldsymbol{x})$ becomes linear in $\boldsymbol{t}$. Moreover, $h_{\boldsymbol{a},b}(\boldsymbol{t}) \mod 2 = m \cdot m'$.

Relinearization makes our public key encryption scheme become a somewhat homomorphic scheme. The secret key is somewhat larger, but as this is only used in decryption, this does not increase the communication or homomorphic evaluation complexity.

3.3.3.2 Modulus Switching.

We have shown how to use relinearization to make a homomorphic evaluation function with the LWE-based encryption scheme. To make this scheme fully homomorphic, the challenge is to manage the noise. One such method is *modulus switching* [BV11]. Essentially, this method changes the ciphertext $c \in \mathbb{Z}_q^n$ to a ciphertext $c' \in \mathbb{Z}_p^n$, where decrypting c' still gives m. This method will be discussed in more detail when discussing the [BGV12] scheme.

3.3.4 Analysis

The performance of the [BV11] scheme is as follows:

- Secret key: The secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$, with size $n \log q = O(\lambda \log \lambda)$ bits.
- Single ciphertext: The ciphertext $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$, with size $(n+1)\log q = O(\lambda \log \lambda)$ bits.
- Public key: $(n+1)((n+1)\log q + 2\lambda)\log q$ bits.
- Evaluation key: $\tilde{O}(n^{2+2\epsilon})$ bits.
- Per-gate computation: $\tilde{O}(k^3 \cdot L^5)$,

where λ is the security parameter, $\epsilon \in (0,1), q = 2^{n^\epsilon}$, $p = 16nk \log 2q$, and L is the maximum depth that the scheme can correctly evaluate circuits.

3.4 Brakerski, Gentry and Vaikuntanathan's Scheme [BGV12]

3.4.1 Overview

The scheme uses a technique named key switching / modulus reduction which generalises the relinearization method we have seen in [BV11]. Here the relinearization procedure can be used to transform any ciphertext $\boldsymbol{c_1}$ decryptable to m with secret key $\boldsymbol{s_1}$ into a ciphertext $\boldsymbol{c_2}$ decryptable to m with secret key $\boldsymbol{s_2}$, not necessarily reducing the dimension of the ciphertext. This enables the evaluation function to be somewhat homomorphic.

To achieve a fully homomorphic scheme, the modulus switching method from [BV11] is again introduced. However, the technique is refined to manage the noise better, so that a fully homomorphic scheme can be achieved without bootstrapping. Bootstrapping is later introduced, but as an optimization technique.

The most significant development of [BGV12] compared to [BV11] is the use of well-known security assumptions based on RLWE, where the use of RLWE over standard LWE paves way to a more efficient fully homomorphic scheme. Also, a careful use of modulus switching achieves fully homomorphic encryption without the need for bootstrapping.

3.4.2 Encryption Scheme

Brakerski, Gentry and Vaikuntanathan use general encryption scheme that can be instantiated to both LWE and RLWE. However, we will focus on the more efficient RLWE setting. The RLWE-based public key encryption scheme as follows. Given the security parameter λ and an additional parameter μ, first choose a μ-bit modulus q. Then choose $d = d(\lambda, \mu)$, $\chi = \chi(\lambda, \mu)$, $n = \lceil 3 \log q \rceil$.

Let $R_q = \mathbb{Z}_q[x]/(f(x))$ with $f(x)$ a polynomial of degree d. To get the secret key, we first draw $\boldsymbol{s'}$ uniformly from χ. The secret key is then $\boldsymbol{s} = (1, \boldsymbol{s'}) \in R_q^2$. To get the public key, first generate vectors $\boldsymbol{A'} \leftarrow R_q^n$, $\boldsymbol{e} \leftarrow \chi^n$, then set $\boldsymbol{b} = -\boldsymbol{A's'} + 2\boldsymbol{e}$. Set the public key $\boldsymbol{A} = (\boldsymbol{b}|\boldsymbol{A'}) \in R_q^{n \times 2}$. Note that $\boldsymbol{A} \cdot \boldsymbol{s} = 2\boldsymbol{e}$.

Suppose $m \in \{0, 1\}$ is the bit we want to encrypt. To encrypt, we do the following:

1. Select a random $\boldsymbol{r} \in R_2^n$ and expand the message to $\boldsymbol{m} = (m, 0) \in R_q^2$.
2. Output $\boldsymbol{c} = \boldsymbol{m} + \boldsymbol{A}^T \boldsymbol{r} \in R_q^2$.

According to $\text{RLWE}_{d,q,\chi}$ (where χ is a uniform distribution over R_q), we can use this scheme a polynomial number of times with negligible probability that an adversary can guess $\boldsymbol{s}$.
To decrypt, we do the following:

1. Compute $b' = [\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q$.
2. Output $m = [b']_2$.

3.4.3 Ideas

3.4.3.1 Key Switching (Dimension Reduction).

This technique generalizes the relinearization method we have seen in [BV11]. It consists of two basic operations as follows:

- $BitDecomp(\boldsymbol{x} \in R_q^n, q)$ decomposes $\boldsymbol{x}$ into its bit representation $\boldsymbol{u} \in R_2^{n\cdot\lceil \log q\rceil}$. We do this by first writing $\boldsymbol{x} = \sum_{i=0}^{\lceil \log q\rceil} 2^i \cdot \boldsymbol{u}_i$ with all $\boldsymbol{u}_i \in R_2^n$, then output $u = (\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{\lceil \log q\rceil}) \in R_2^{n\cdot\lceil \log q\rceil}$.
- $Powersof2(\boldsymbol{x} \in R_q^n, q)$ expands $\boldsymbol{x}$ into $\boldsymbol{u} \in R_q^{n\cdot\lceil \log q\rceil}$ that has copies of $\boldsymbol{x}$ multiplied by powers of 2. The output is $(\boldsymbol{x}, 2\cdot\boldsymbol{x}, \cdots, 2^{\lceil \log q\rceil}\boldsymbol{x}) \in R_q^{n\cdot\lceil \log q\rceil}$.

Lemma 3.4.1 $\langle BitDecomp(\boldsymbol{c}, q), Powersof2(s, q)\rangle = \langle \boldsymbol{c}, s\rangle \mod q$.

Proof We will give a more detailed proof than in [BV11]. Writing $\boldsymbol{c} = \sum_{i=0}^{\lceil \log q\rceil} 2^i \cdot \boldsymbol{c}_i$, we have $BitDecomp(\boldsymbol{c}, q) = (\boldsymbol{c}_0, \boldsymbol{c}_1, \cdots, \boldsymbol{c}_{\lceil \log q\rceil})$. Also, $Powersof2(s, q) = (\boldsymbol{s}, 2\cdot\boldsymbol{s}, \cdots, 2^{\lceil \log q\rceil}\boldsymbol{s})$. Hence,

$$\begin{aligned}
\langle BitDecomp(\boldsymbol{c}, q), Powersof2(s, q)\rangle &= \sum_{i=0}^{\lceil \log q\rceil} \langle \boldsymbol{c}_i, 2^i \cdot \boldsymbol{s}\rangle \\
&= \sum_{i=0}^{\lceil \log q\rceil} 2^i \cdot \langle \boldsymbol{c}_i, \boldsymbol{s}\rangle \\
&= \sum_{i=0}^{\lceil \log q\rceil} \langle 2^i \cdot \boldsymbol{c}_i, \boldsymbol{s}\rangle \\
&= \langle \sum_{i=0}^{\lceil \log q\rceil} 2^i \cdot \boldsymbol{c}_i, \boldsymbol{s}\rangle \\
&= \langle \boldsymbol{c}, s\rangle \mod q
\end{aligned}$$

∎

The key switching technique can be defined by the following two operations. $SwitchKeyGen(s_1 \in R_q^{n_1}, s_2 \in R_q^{n_2})$:

1. Generate a public key $\boldsymbol{A}$ as previously described, but with secret key s_2 and parameter $n = n_1 \cdot \lceil \log q\rceil$.
2. Set $\boldsymbol{B} = [Powersof2(s_1)|O]$, that is the matrix with first column containing $Powersof2(s_1)$ and augmenting some columns with all elements zero until it matches the size of $\boldsymbol{A}$.
3. Set $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$, and output $\tau_{s_1\to s_2} = \boldsymbol{C}$.

$SwitchKey(\tau_{s_1 \to s_2}, c_1)$: Output $c_2 = BitDecomp(c_1)^T \cdot \boldsymbol{C}$.

The following lemma proves that key switching works.

Lemma 3.4.2 *Let $s_1, s_2, q, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ be as in $SwitchKeyGen(s_1, s_2)$, and let $\boldsymbol{A} \cdot s_2 = 2e_2 \in R_q^N$. Let $c_1 \in R_q^{n_1}$ and $c_2 \leftarrow SwitchKey(\tau_{s_1 \to s_2}, c_1)$. Then we have*

$$\langle c_2, s_2 \rangle = 2\langle BitDecomp(c_1), e_2 \rangle + \langle c_1, s_1 \rangle \mod q.$$

Proof We will give a more detailed proof than in [BV11]. By definition,

$$\begin{aligned}
\langle c_2, s_2 \rangle &= \langle BitDecomp(c_1)^T \cdot \boldsymbol{C}, s_2 \rangle \\
&= BitDecomp(c_1)^T \cdot \boldsymbol{C} \cdot s_2 \\
&= BitDecomp(c_1)^T \cdot (\boldsymbol{A} + \boldsymbol{B}) \cdot s_2 \\
&= BitDecomp(c_1)^T \cdot (2e_2 + Powersof2(s_1)) \\
&= 2\langle BitDecomp(c_1), e_2 \rangle + \langle BitDecomp(c_1), Powersof2(s_1) \rangle \\
&= 2\langle BitDecomp(c_1), e_2 \rangle + \langle c_1, s_1 \rangle \mod q. \text{ (from Lemma 3.4.1)}
\end{aligned}$$

∎

This lemma implies that key switching only produces an error $\|2\langle BitDecomp(c_1), e_2 \rangle\|_2$ which is small because $BitDecomp(c_1)$ only has coefficients 0 or 1 in the inner product.

3.4.3.2 Modulus Switching.

The modulus switching technique used is a variant of the one used in [BV11]. This method changes the ciphertext $c \in R_q^2$ to a ciphertext $c' \in R_p^2$, where decrypting c' still gives m.

Suppose we have a ciphertext $\boldsymbol{c} = (c_1, c_2) \in R_q^2$, and consider $\boldsymbol{c'}$ to be the vector closest (using ℓ_1-norm) to $(p/q) \cdot \boldsymbol{c}$. such that $\boldsymbol{c'} \equiv \boldsymbol{c} \mod 2$. Note that for some $k \in \mathbb{Z}$ we have $[\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q = \langle \boldsymbol{c}, \boldsymbol{s} \rangle - kq$. Define $e = \langle \boldsymbol{c'}, \boldsymbol{s} \rangle - kp = [\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q + (\langle \boldsymbol{c'}, \boldsymbol{s} \rangle - \langle \boldsymbol{c'}, \boldsymbol{s} \rangle) + (kq - kp)$. So $e \equiv [\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q \mod 2$. Also, if s is chosen such that $|[\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q| < q/2 - (q/p)\ell_1(\boldsymbol{s})$, we can then show that $e \equiv [\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_p \mod 2$, which means that decrypting $\boldsymbol{c}$ and $\boldsymbol{c'}$ will result to the same message. Moreover with this choice of s, we can also show that $|[\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_p| < |[\langle \boldsymbol{c}, \boldsymbol{s} \rangle]_q| + \ell_1(\boldsymbol{s})$. This means that if we choose a short secret key s (i.e. with a small $\ell_1(\boldsymbol{s})$) and small enough p relative to q, we can significantly decrease the amount of noise in the ciphertext.

Now we will give a short analysis (slightly modified from [BGV12]) of how much noise can actually be reduced. Suppose q is approximately x^k, and we have two ciphertexts with noise approximately x. Without modulus switching, note that addition creates noise of size $2x$, and multiplication creates noise of size x^2. Hence, we can evaluate multiplication with depth at most $\log k$ before the noise becomes too large. However, using modulus switching, we get that the noise after multiplication comes down from x^2 back to x, with the modulus reduced from q_i to q_i/x. So by choosing a good chain of

decreasing moduli $(q, q/x, q/x^2, \cdots)$, there can be up to k levels of multiplication. Also, this method can be used at any time during evaluation because we did not require the secret key to perform it.

3.4.4 Analysis

The performance of the [BGV12] scheme is as follows (RLWE case):

- Secret key: The secret key is 2 ring elements, which require $2d \log q$ bits.
- Single ciphertext: The ciphertext also consists of 2 ring elements, which require $2d \log q$ bits.
- Public key: The public key $\boldsymbol{A} \in R_q^{n \times 2}$ consists of $2n$ ring elements, which require $2dn \log q$ bits.
- Key switching: $\tilde{O}(dn^3 \log^2 q)$.
- Modulus switching: $\tilde{O}(dn^2 \log q)$.
- Per-gate computation: $\tilde{O}(k \cdot L^3)$,

where, with λ as the security parameter, $q = \Theta(2^\lambda)$, $d = \Omega(\lambda \log \lambda)$, $n = \lceil 3 \log q \rceil$, and L is the maximum depth that the scheme can correctly evaluate circuits. Here, we recall that $R_q = \mathbb{Z}_q[x]/(f(x))$ with $f(x)$ a polynomial of degree d. So an element in R_q has size $\log q^d = d \log q$ bits. Also, we focus the analysis on the RLWE version, as it is more efficient than the LWE instantiation.

3.5 Brakerski's Scheme [Bra12]

3.5.1 Overview

The scheme works in an invariant perspective, where only the ratio q/B matters. This is done by scaling the ciphertext down by a factor of q (that is, $c' = c/q$). In this perspective, homomorphic multiplication multiplies the noise by a polynomial factor $p(n)$, which is an improvement from [BGV12] where homomorphic multiplication squares the noise. One significant change in this scheme is that it does not use modulus switching as in the previous two schemes.

3.5.2 Encryption Scheme

Brakerski uses Regev's LWE-based public key encryption scheme [Reg05] as follows. Given the security parameter n, let $q = q(n)$ be an integer and $\chi = \chi(n)$ be a distribution over $\mathbb{Z}$. We have the secret key $\boldsymbol{s} = (s[1], \cdots, s[n]) \in \mathbb{Z}_q^n$. To get the public key, first let $N = (n+1) \cdot (\log q + O(1))$. Sample $\boldsymbol{A} \leftarrow \mathbb{Z}^{N \times n}$ and $\boldsymbol{e} \leftarrow \chi^N$. Compute $\boldsymbol{b} = [\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}]_q$. The public key is then $\boldsymbol{P} = [\boldsymbol{b} | -\boldsymbol{A}] \in \mathbb{Z}^{N \times (n+1)}$.

Suppose $m \in \{0, 1\}$. To encrypt m, we do the following:

1. Select a random $\boldsymbol{r} \in \{0, 1\}^N$.

2. Set $\boldsymbol{m} = (m, 0, \cdots, 0) \in \{0,1\}^{n+1}$.
3. Output $\boldsymbol{c} = [\boldsymbol{P^T} \cdot \boldsymbol{r} + \lfloor q/2 \rfloor \cdot \boldsymbol{m}]_q \in \mathbb{Z}_q^{n+1}$

To decrypt a ciphertext $\boldsymbol{c}$, we do the following:

1. Compute $c_0 = [\langle \boldsymbol{c}, (1, \boldsymbol{s}) \rangle]_q$.
2. Output $m = [\lfloor 2 \cdot c_0/q \rceil]_2$.

The correctness of this scheme can be seen from analyzing the encryption and decryption noise. The noise magnitude of properly encrypted ciphertexts can be shown to be small, by the following lemma.

Lemma 3.5.1 *Let $q, n, N, |\chi| \leq B$ be parameters for Regev's public key encryption scheme. Let $\boldsymbol{s} \in \mathbb{Z}^n$ be a vector and $m \in \{0,1\}$ be some bit. Set $\boldsymbol{P}$ as the public key generated from Regev's scheme with secret key $\boldsymbol{s}$, and $\boldsymbol{c}$ be the ciphertext created by encrypting m under public key $\boldsymbol{P}$. Then for some e with $|e| \leq N \cdot B$ it holds that*

$$\langle \boldsymbol{c}, (1, \boldsymbol{s}) \rangle = \lfloor q/2 \rfloor \cdot m + e \pmod{q}$$

Proof We will give a detailed and more elementary proof than in [Bra12]. Let $\boldsymbol{r}$ be the random element sampled, and $\boldsymbol{m}$ be the extended message vector in the Regev encryption. Then we have that

$$\begin{aligned}
\langle \boldsymbol{c}, (1, \boldsymbol{s}) \rangle &= \langle \boldsymbol{P}^T \cdot \boldsymbol{r} + \lfloor q/2 \rfloor \cdot \boldsymbol{m}, (1, \boldsymbol{s}) \rangle \\
&= \langle \lfloor q/2 \rfloor \cdot \boldsymbol{m}, (1, \boldsymbol{s}) \rangle + \langle \boldsymbol{P}^T \cdot \boldsymbol{r}, (1, \boldsymbol{s}) \rangle \\
&= \lfloor q/2 \rfloor \cdot m + \boldsymbol{r}^T \boldsymbol{P} \cdot (1, \boldsymbol{s}) \\
&= \lfloor q/2 \rfloor \cdot m + \boldsymbol{r}^T (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{s}) \\
&= \lfloor q/2 \rfloor \cdot m + \boldsymbol{r}^T \cdot \boldsymbol{e} \\
&= \lfloor q/2 \rfloor \cdot m + \langle \boldsymbol{r}, \boldsymbol{e} \rangle \pmod{q}.
\end{aligned}$$

As $r \in \{0,1\}^N$, we have $|\boldsymbol{r}| \leq N$. Also, by definition $|\boldsymbol{e}| \leq B$. So by Cauchy-Schwarz, we have that $|\langle \boldsymbol{r}, \boldsymbol{e} \rangle| \leq |\boldsymbol{r}||\boldsymbol{e}| \leq N \cdot B$, and the lemma follows by setting $e = \langle \boldsymbol{r}, \boldsymbol{e} \rangle$. ∎

Moreover, for ciphertexts with a small noise, decryption gives the correct message according to the following lemma:

Lemma 3.5.2 *Let $\boldsymbol{s} \in \mathbb{Z}^n$ be some vector, and let $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$ be such that*

$$\langle \boldsymbol{c}, (1, \boldsymbol{s}) \rangle = \lfloor q/2 \rfloor \cdot m + e \pmod{q}$$

with $m \in \{0,1\}$ and $|e| \leq q/4$. Then the decryption of $\boldsymbol{c}$ under secret key $\boldsymbol{s}$ outputs m.

Proof By definition, we have that in the decryption, $c_0 = [\langle \boldsymbol{c}, (1, \boldsymbol{s})\rangle]_q = \lfloor q/2 \rfloor \cdot m + e$. So, decryption outputs

$$\begin{aligned} [\lfloor 2 \cdot c_0/q \rceil]_2 &= [\lfloor 2 \cdot (\lfloor q/2 \rfloor \cdot m + e)/q \rceil]_2 \\ &= [\lfloor 2\lfloor q/2 \rfloor \cdot m/q + 2e/q \rceil]_2 \\ &= m \end{aligned}$$

since $|2e/q| < 2(q/4)/q = 1/2$ so the rounding is correct. ∎

3.5.3 Ideas

3.5.3.1 Homomorphic properties.

We start from Regev's public key encryption scheme, where the encryption of $m \in \{0, 1\}$ is a vector $\boldsymbol{c} \in \mathbb{Z}_q^n$ such that $[\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_q = \lfloor \frac{q}{2} \rfloor \cdot m + e$ with $|e| \leq E$. First take the invariant perspective, and set $\boldsymbol{c'} = \boldsymbol{c}/q$. Then $[\langle \boldsymbol{c'}, \boldsymbol{s}\rangle]_1 = \frac{1}{2} \cdot m + e'$ with $|e'| \leq E/q = \epsilon$. Additive homomorphism can be seen directly in this perspective: if $\boldsymbol{c_1}, \boldsymbol{c_2}$ encrypt m_1, m_2 respectively, then

$$\boldsymbol{c}_{add} = \boldsymbol{c_1} + \boldsymbol{c_2}$$

encrypts $[m_1 + m_2]_2$, with noise approximately 2ϵ. Multiplicative homomorphism is done by defining

$$\boldsymbol{c}_{mult} = 2 \cdot \boldsymbol{c_1} \otimes \boldsymbol{c_2}.$$

The above tensored ciphertext can be decrypted using a tensored secret key $\boldsymbol{s} \otimes \boldsymbol{s}$, because

$$\langle 2 \cdot \boldsymbol{c_1} \otimes \boldsymbol{c_2}, \boldsymbol{s} \otimes \boldsymbol{s}\rangle = 2\langle \boldsymbol{c_1}, \boldsymbol{s}\rangle \cdot \langle \boldsymbol{c_2}, \boldsymbol{s}\rangle.$$

To show why this definition works, we have to show that $[2\langle \boldsymbol{c_1}, \boldsymbol{s}\rangle \cdot \langle \boldsymbol{c_2}, \boldsymbol{s}\rangle]_1 \approx \frac{1}{2} m_1 m_2 + e'$, for a small e'. We start by letting $I_1, I_2 \in \mathbb{Z}$ be integers, and e_1, e_2 with absolute value less than ϵ be rational numbers such that

$$\begin{aligned} \langle \boldsymbol{c_1}, \boldsymbol{s}\rangle &= \frac{1}{2} m_1 + e_1 + I_1 \\ \langle \boldsymbol{c_2}, \boldsymbol{s}\rangle &= \frac{1}{2} m_2 + e_2 + I_2 \end{aligned}$$

Then we have:

$$\begin{aligned} 2\langle \boldsymbol{c_1}, \boldsymbol{s}\rangle \cdot \langle \boldsymbol{c_2}, \boldsymbol{s}\rangle &= 2 \cdot (\frac{1}{2} m_1 + e_1 + I_1) \cdot (\frac{1}{2} m_1 + e_2 + I_2) \\ &= \frac{1}{2} m_1 m_2 + 2(e_1 I_2 + e_2 I_1) + (e_1 m_2 + e_2 m_1 + 2e_1 e_2) \\ &\quad + (m_1 I_2 + m_2 I_1 + 2 I_1 I_2) \end{aligned}$$

But $m_1, m_2 \in \{0, 1\}$ so $m_1 I_2 + m_2 I_1 + 2 I_1 I_2 \in \mathbb{Z}$. Also, the term $2e_1 e_2$ that squares the noise in [BV11] and [BGV12] can now be ignored as $|2e_1 e_2| \leq 2\epsilon^2 \ll \epsilon$. By the triangle inequality, we also have that $|e_1 m_2 + e_2 m_1| \leq |e_1 m_2| + |e_2 m_1| \leq |e_1| + |e_2| < 2\epsilon$. Therefore the noise is dominated by the term $e' = 2(e_1 I_2 + e_2 I_1)$. [Bra12] shows that

this term is bounded by $O(\|\boldsymbol{s}\|_1) \cdot \epsilon$, and that by choosing $q \leq 2^n$, $\|\boldsymbol{s}\|_1$ only depends on n and independent of B, q. Hence we have:

$$\begin{aligned}[2\langle \boldsymbol{c_1}, \boldsymbol{s}\rangle \cdot \langle \boldsymbol{c_2}, \boldsymbol{s}\rangle]_1 &= \frac{1}{2}m_1m_2 + 2(e_1I_2 + e_2I_1) + (e_1m_2 + e_2m_1 + 2e_1e_2) \\ &\approx \frac{1}{2}m_1m_2 + 2(e_1I_2 + e_2I_1) \\ &\approx \frac{1}{2}m_1m_2 + e'\end{aligned}$$

3.5.3.2 Vector Decomposition and Key Switching.

Vector decomposition is done to reduce the norm of $\boldsymbol{s}$, resulting in the previous discussion having a smaller noise. Initially, $\|\boldsymbol{s}\|_1 \leq n \cdot q$, as the elements of the secret key $\boldsymbol{s}$ are sampled uniformly from $\mathbb{Z}_{\|}$. As in [BGV12], vector decomposition uses two basic operations $BitDecomp$ and $Powersof2$.

Key switching is done as in Brakerski, Gentry and Vaikuntanathan's scheme. It also uses the operations $BitDecomp$ and $Powersof2$ to define $SwitchKeyGen(s_1, s_2)$ and $SwitchKey(\tau_{s_1 \to s_2}, c_1)$.

3.5.4 Analysis

The performance of the [Bra12] scheme is as follows:

- Secret key: $\boldsymbol{s} \in \mathbb{Z}_q^n$, with size $n \log q$ bits.
- Single ciphertext: $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$, with size $(n+1)\log q$ bits.
- Public key: $\boldsymbol{P} \in \mathbb{Z}_q^{N \times n}$, with size $N \cdot n \log q$ bits. But $N = O(n \log q)$, so $\boldsymbol{P}$ has size $O(n^2 \log^2 q)$ bits.

3.6 Fan and Vercauteren's Scheme [FV12]

3.6.1 Overview

This scheme improves upon Brakerski's scheme by using a more efficient scheme that bases its assumptions on RLWE instead of LWE. Specifically, it contains a modified version of the LPR scheme for optimization and easier analysis. Also, there will be a relinearization process similar to that discussed in Brakerski, Gentry and Vaikuntanathan's scheme, so there will be a need to have an additional element to the LPR scheme which is the relinearization key rlk. This relinearization key will be used to compute the homomorphic multiplication $\boldsymbol{c}_{mult}$.

3.6.2 Encryption Scheme

Fan and Vercauteren uses the RLWE-based LPR scheme as follows [LPR10].

- $R = \mathbb{Z}[x]/(x^d + 1)$, where d is a power of 2, and set the message space to be R_t for some integer $t > 1$. Set $\Delta = \lfloor q/t \rfloor$.

- Secret key: $s \in R_q$, sampled from a noise distribution χ.
- Public key: $(b = -(a \cdot s + e) \mod q, a) \in R_q^2$, where a is sampled from R_q
- Encryption: Suppose we want to encrypt $m \in R_t$.
  Sample r, e_1, e_2 from χ. Return (u, v) where

$$\begin{aligned} u &= a \cdot r + e_1 + \Delta \cdot m \mod q, \\ v &= b \cdot r + e_2 \mod q. \end{aligned}$$

- Decryption: First compute $u + v \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \Delta \cdot m \mod q$. Then multiply by $\frac{t}{q}$ and round to the nearest integer modulo t.

As in Brakerski's scheme, we can show that decryption is correct for properly encrypted ciphertexts. This is dealt with by the following lemma. One important thing to note is that all norms used in this scheme is the infinity norm $\|.\|_\infty$, not the Euclidean norm as in the previous schemes. The expansion factor also uses the infinity norm:

$$\delta_R = \max\{\frac{\|a \cdot b\|_\infty}{\|a\|_\infty \|b\|_\infty} : a, b \in R\}.$$

With this in mind, we can proceed to the lemma:

Lemma 3.6.1 *If $||\chi||_\infty < B$, then for some $||v||_\infty \leq 2 \cdot \delta_R \cdot B^2 + B$ we have that*

$$[u + v \cdot s]_q = \Delta \cdot \boldsymbol{m} + \boldsymbol{v}. \tag{3.1}$$

Moreover, if $2 \cdot \delta_R \cdot B^2 + B < \Delta/2$, decryption works correctly.

.

Proof We will give a partial proof, which is more detailed, but use a claim in [FV12]. Using the definitions from the encryption, we have

$$\begin{aligned} u + v \cdot s &= a \cdot r + e_1 + \Delta \cdot m + b \cdot r \cdot s + e_2 \cdot s \\ &= \Delta \cdot m + (e \cdot u + e_1 + e_2 \cdot s) \mod q. \end{aligned}$$

If we set $\boldsymbol{v} = e \cdot r + e_1 + e_2 \cdot s$, then as $||x \cdot y||_\infty \leq \|x\|_\infty \cdot \|y\|_\infty \delta_R \leq B^2 \delta_R$, we have

$$\begin{aligned} \boldsymbol{v} &\leq \|e \cdot r\|_\infty + \|e_1\|_\infty + \|e_2 \cdot s\|_\infty \\ &\leq \delta_R \cdot B^2 + B + \delta_R \cdot B^2 \\ &= 2 \cdot \delta_R \cdot B^2 + B. \end{aligned}$$

Let $\boldsymbol{r}$ be an element such that $\boldsymbol{u} + v \cdot \boldsymbol{s} = \Delta \cdot m + v + q \cdot \boldsymbol{r}$. Then we have

$$\begin{aligned} \frac{t}{q}(\boldsymbol{u} + v \cdot \boldsymbol{s}) &= t/q \cdot \Delta \cdot m + (t/q) \cdot v + t \cdot \boldsymbol{r} \\ &= t/q \cdot (q/t - \epsilon) \cdot m + (t/q) \cdot v + t \cdot \boldsymbol{r} \\ &= m + (t/q) \cdot (v - \epsilon \cdot m) + t \cdot \boldsymbol{r}, \end{aligned}$$

where $\epsilon = q/t - \Delta = q/t - \lfloor q/t \rfloor < 1$.
Fan and Vercauteren claim that $(t/q) \cdot \|v - \epsilon \cdot m\|_\infty < 1/2$ [FV12]. With this claim, and noting that $m \in R_t$, $m + (t/q) \cdot (v - \epsilon \cdot m) + t \cdot \boldsymbol{r} \mod t = m + (t/q) \cdot (v - \epsilon \cdot m)$ rounds to m. ∎

3.6.3 Ideas

The encryption scheme is a modified version of the LPR scheme, where the s, u are sampled from $\mathbb{R}_2$ instead of χ. Fan and Vercauteren argue that assuming the results for the LWE setting carry over to the RLWE setting, this modification will have minor security implications. This will imply $\|s\|_\infty = \|r\|_\infty = 1$, and the bound in the previous lemma becomes

$$\begin{aligned} \boldsymbol{v} &\leq \delta_R \|e\|_\infty \cdot \|r\|_\infty + \|e_1\|_\infty + \delta_R \|e_2\|_\infty \cdot \|s\|_\infty \\ &\leq \delta_R \cdot B \cdot 1 + B + \delta_R \cdot B \cdot 1 \\ &= 2 \cdot \delta_R \cdot B + B. \end{aligned}$$

The main invariant is given in (3.1), where if we interpret the elements of the ciphertext as the coefficients of the polynomial $ct(x)$ (that is, $ct(x) = u + v \cdot s$), then evaluating this polynomial with $x = s$ will give us:

$$[ct(s)]_q = \Delta \cdot m + v,$$

which using the previous lemma enables us to correctly recover the message m.

3.6.3.1 Homomorphic Properties

3.6.3.1.1 Additive homomorphism. An appropriate operation for additive homomorphism can be seen directly. First note that if $[ct_i(\boldsymbol{s})]_q = \Delta \cdot \boldsymbol{m}_i + v_i$, then we have that

$$[ct_1(\boldsymbol{s}) + ct_2(\boldsymbol{s})]_q = \Delta \cdot [\boldsymbol{m}_1 + \boldsymbol{m}_2]_t + \boldsymbol{v}_1 + \boldsymbol{v_2} - \epsilon \cdot t \cdot \boldsymbol{r}$$

where $\epsilon = q/t - \Delta = q/t - \lfloor q/t \rfloor < 1$. Moreover, using the modified version, $\|\boldsymbol{r}\|_\infty \leq 1$, so the noise grows additively with maximum $\|t \cdot \boldsymbol{r}\|_\infty \leq t$. So we can define:

$$\boldsymbol{c}_{add}(ct_1, ct_2) = ([ct_1[0] + ct_2[0]]_q, [ct_1[1] + ct_2[1]]_q)$$

3.6.3.1.2 Multiplicative homomorphism. Finding an operation for multiplicative homomorphism is not so straightforward. First we define

$$ct_i = \Delta \cdot \boldsymbol{m}_i + v_i + q \cdot r_i.$$

Then multiplying for $i = 1, 2$ gives us:

$$\begin{aligned} (ct_1 \cdot ct_2)(s) &= (\Delta \cdot \boldsymbol{m}_1 + v_1 + q \cdot r_1) \cdot (\Delta \cdot \boldsymbol{m}_2 + v_2 + q \cdot r_2) \\ &= \Delta^2 \cdot m_1 \cdot m_2 + \Delta \cdot (m_1 \cdot v_2 + m_2 \cdot v_1) + q(v_1 \cdot r_2 + v_2 \cdot r_1) \\ &+ v_1 \cdot v_2 + q \cdot \Delta \cdot (m_1 \cdot r_2 + m_2 \cdot r_1) + q^2 \cdot r_1 \cdot r_2. \end{aligned}$$

We can see that to get an encryption of $[m_1 \cdot m_2]_t$, we must divide the above equation by Δ. However, this might create errors in rounding, as Δ does not necessarily divide q.

To prevent rounding errors, we instead divide by q/t (or equivalently multiply by t/q). Let $ct_1(x) + ct_2(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2$. Then we can get the approximation:

$$\frac{t}{q} \cdot (ct_1 \cdot ct_2)(s) = \lfloor t \cdot \boldsymbol{c}_0/q \rceil + \lfloor t \cdot \boldsymbol{c}_1/q \rceil \cdot \boldsymbol{s} + \lfloor t \cdot \boldsymbol{c}_2/q \rceil \cdot \boldsymbol{s}^2 + \boldsymbol{r}_a$$

where

$$\boldsymbol{r}_a = (\lfloor t \cdot \boldsymbol{c}_0/q \rceil - t \cdot \boldsymbol{c}_0/q) + (\lfloor t \cdot \boldsymbol{c}_1/q \rceil - t \cdot \boldsymbol{c}_1/q) \cdot \boldsymbol{s} + (\lfloor t \cdot \boldsymbol{c}_2/q \rceil - t \cdot \boldsymbol{c}_2/q) \cdot \boldsymbol{s}^2$$

and by the triangle inequality and the fact that $\|a - \lfloor a \rceil\|_\infty \leq 1/2$ for all real numbers a, we have

$$\begin{aligned}
\|\boldsymbol{r}_a\|_\infty &\leq \|\lfloor t \cdot \boldsymbol{c}_0/q \rceil - t \cdot \boldsymbol{c}_0/q\|_\infty + \|(\lfloor t \cdot \boldsymbol{c}_1/q \rceil - t \cdot \boldsymbol{c}_1/q) \cdot \boldsymbol{s}\|_\infty \\
&\quad + \|(\lfloor t \cdot \boldsymbol{c}_2/q \rceil - t \cdot \boldsymbol{c}_2/q) \cdot \boldsymbol{s}^2\|_\infty \\
&\leq 1/2 + 1/2 \cdot \delta_R \cdot \|s\|_\infty + 1/2 \cdot \delta_R^2 \cdot \|s\|_\infty^2 \\
&< 1/2 + \cdot \delta_R \cdot \|s\|_\infty + 1/2 \cdot \delta_R^2 \cdot \|s\|_\infty^2 \\
&= (\delta_R \cdot \|s\|_\infty + 1)^2/2.
\end{aligned}$$

This gives an idea of a homomorphic multiplication where ct_1, ct_2 each with two elements is multiplied into a result with three elements.

$$\boldsymbol{c}_{basicmult}(ct_1, ct_2) = \frac{t}{q} \cdot (ct_1 \cdot ct_2)(s)$$

Fan and Vercauteren analyze the noise using the following lemma [FV12].

Lemma 3.6.2 *Let ct_i for $i = 1, 2$ be two ciphertexts, with $[ct_i(\boldsymbol{s})]_q = \Delta \cdot \boldsymbol{m}_i + v_i$ and E such that $\|v_i\|_\infty < E < \Delta/2$. Let $ct_1(x) + ct_2(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2$. Then*

$$[\lfloor t \cdot \boldsymbol{c}_0/q \rceil + \lfloor t \cdot \boldsymbol{c}_0/q \rceil \cdot \boldsymbol{s} + \lfloor t \cdot \boldsymbol{c}_2/q \rceil \cdot \boldsymbol{s}^2]_q = \Delta \cdot [m_1 m_2]_t + \boldsymbol{v}_3$$

with $\|v_3\|_\infty < 2 \cdot \delta_R \cdot t \cdot E \cdot (\delta_R \cdot \|s\|_\infty + 1) + 2 \cdot t^2 \cdot \delta_R^2 \cdot (\|s\|_\infty + 1)^2$.

By using this lemma, and noting that the term $2 \cdot \delta_R \cdot t \cdot E \cdot (\delta_R \cdot \|s\|_\infty + 1) + 2 \cdot t^2 \cdot \delta_R^2 \cdot (\|s\|_\infty + 1)^2$ is dominated by $2 \cdot t^2 \cdot \delta_R^2 \cdot \|s\|_\infty^2$, we can see that the noise is multiplied roughly by $2 \cdot t \cdot \delta_R^2 \cdot \|s\|_\infty$. Using the optimization stated before with $\|s\|_\infty = 1$, the noise is multiplied roughly by a much smaller factor $2 \cdot t \cdot \delta_R^2$ after multiplication.

3.6.3.2 Relinearization.

The previous lemma shows that we can do multiplication, at the cost of increasing the size of the ciphertext. To keep the number of ciphertext elements down, Fan and Vercauteren use relinearization like in the previous schemes. The goal is to transform a degree 2 ciphertext we obtained from basic multiplication, $ct = [\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2]$ into a degree 1 ciphertext $ct' = [\boldsymbol{c}'_0, \boldsymbol{c}'_1]$, such that

$$[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{c}_2 \cdot \boldsymbol{s}^2]_q = [\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s} + \boldsymbol{r}]_q$$

where $\|r\|$ is a small error, meaning that ct and ct' will both correctly decrypt to the same message $m \in R_t$. This step will require a relinearization key rlk. Fan and Vercauteren have two different ideas of relinearization that they propose.

3.6.3.2.1 Relinearization version 1. The first idea is to further generalize the key switching technique in [BGV12], by decomposing into a base T (the previous schemes use $T = 2$). This is done by writing $\boldsymbol{c}_2$ in base T, that is $\boldsymbol{c}_2 = \sum_{i=0}^{l} T^i \cdot c_2^{(i)} \mod q$. Generalizing the relinearization key in the previous schemes, the relinearization key rlk will consist of elements $T^i \boldsymbol{s}^2$ masked with some noise:

$$rlk = [([-(a_i \cdot s + e_i) + T^i \cdot s^2]_q, a_i) : i \in [0...l].$$

Here, the relinearization key rlk uses $l+1$ bits, with $l = \lfloor \log_T q \rceil$. This means that when T increases, $\log_T q = \frac{\log q}{\log T}$ decreases, and hence the relinearization key is smaller.

3.6.3.2.2 Relinearization version 2. The second idea uses some form of modulus switching, by switching from modulo q to modulo $p \cdot q$ for some integer p. The idea here is that it is sufficient to approximate $\boldsymbol{c}_2 \cdot \boldsymbol{s}^2$ modulo q, that is find $c_{2,0}, c_{2,1}$ such that $c_{2,0} + c_{2,0} \cdot s = c_2 \cdot s^2 + r$ for a small r. So the relinearization key is of the form:

$$rlk = ([-(a \cdot s + e) + p \cdot s^2]_{p \cdot q}, a), a \in R_{p \cdot q}, e \leftarrow \chi.$$

3.6.3.3 Redefinition of homomorphic multiplication.

By using one of these two relinearization techniques, we can define homomorphic multiplication such that evaluating a multiplication still results in two elements.
In the first version, we have

$$\boldsymbol{c}_{mult} = (c_0', c_1'),$$

where

$$\begin{aligned} c_0' &= [c_0 + \sum_{i=0}^{l} rlk[i][0] \cdot c_2^{(i)}]_q, \\ c_1' &= [c_1 + \sum_{i=0}^{l} rlk[i][1] \cdot c_2^{(i)}]_q, \end{aligned}$$

and $rlk = [([-(a_i \cdot s + e_i) + T^i \cdot s^2]_q, a_i) : i \in [0...l]$.
In the second version, we have

$$\boldsymbol{c}_{mult} = ([c_0 + c_{2,0}]_q, [c_1 + c_{2,1}]_q),$$

where

$$\begin{aligned} c_{2,0} &= [\lfloor \frac{c_2 \cdot rlk[0]}{p} \rceil]_q, \\ c_{2,1} &= [\lfloor \frac{c_2 \cdot rlk[0]}{p} \rceil]_q, \end{aligned}$$

and $rlk = ([-(a \cdot s + e) + p \cdot s^2]_{p \cdot q}, a)$.

3.6.3.4 Towards Fully Homomorphic Encryption.

The previous idea gives us a somewhat homomorphic encryption scheme, as it can only evaluate functions up to some maximum level before the noise becomes too big. To turn the scheme into a fully homomorphic encryption scheme, Fan and Vercauteren use Gentry's bootstrapping technique. Here, the decryption function of the somewhat homomorphic scheme is evaluated homomorphically to obtain an encryption of the same message as before, but with a smaller noise (according to the depth of the decryption circuit.) Fan and Vercauteren consider two cases: the optimized case which gives the simplest decryption function, and a general case.

3.6.3.4.1 Optimized case: $q = 2^n$ and $t = 2^{n-k}, k > 0$. Then we can write $\Delta = 2^k$, so any division by Δ will be a simple right shift. Also, since

$$\frac{t}{q} \cdot [c_0 + c_1 \cdot s]_q = \frac{[c_0 + c_1 \cdot s]_q}{\Delta},$$

decryption can be done fast.

3.6.3.4.2 General Case. This case is dealt with by reducing to the optimized case by a form of modulus switching. From section 3.6.3.1 and lemma 3.6.1, a ciphertext ct satisfies $ct[0] + ct[1] \cdot \boldsymbol{s} = \Delta \cdot \boldsymbol{m} + \boldsymbol{v} + q \cdot \boldsymbol{r}$, with $\|\boldsymbol{v}\|_\infty < \Delta/2$. Assuming the noise $\boldsymbol{v}$ has not reached is maximal size, we can switch from modulus q to modulus 2^n where $2^n \leq q < 2^{n+1}$ by multiplying the ciphertext by $2^n/q$. So if we set

$$\begin{aligned} c_0 &= \lfloor 2^n \cdot ct[0]/q \rceil, \\ c_1 &= \lfloor 2^n \cdot ct[1]/q \rceil, \end{aligned}$$

and note that $\frac{2^n}{q} \cdot \frac{q}{t} = \frac{2^n}{t}$ and $\frac{2^n}{q} \cdot q = 2^n$, we get that

$$\begin{aligned} c_0 + c_1 \cdot s &= \lfloor 2^n \cdot ct[0]/q \rceil + \lfloor 2^n \cdot ct[1]/q \rceil \cdot \boldsymbol{s} \\ &= \lfloor \frac{2^n}{t} \rfloor m + \boldsymbol{e} + 2^n \cdot \boldsymbol{r}. \end{aligned}$$

As long as the new error $\|\boldsymbol{e}\|_\infty < \lfloor 2^n/t \rfloor/2$, lemma 3.6.1 says we will now obtain a valid ciphertext modulus 2^n. By considering (c_0, c_1) as the ciphertext to decrypt, decryption now becomes as simple as the optimized case.

3.6.4 Analysis

The performance of the [FV12] scheme is as follows:

- Secret key: The secret key $\boldsymbol{s}$ is sampled from a distribution χ over R, so its size will be a function of d and λ. With the optimization that $\boldsymbol{s}$ is sampled from R_2, we have that the secret key is d bits.
- Single ciphertext: The ciphertext is two elements in R_q, so it has size $2d \log q$ bits.
- Public key: The public key is also two elements in R_q, so it has size $2d \log q$ bits.

- Relinearization version 1:
  - Relinearization key: As discussed in Section 3.6.3.2, the relinearization key rlk uses $l+1$ bits, with $l = \lfloor \log_T q \rceil$. So the relinearization key is approximately $\log_T q$ bits.
  - Number of operations: The formula shown in Section 3.6.3.2 shows that $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ each do $l = \lfloor \log_T q \rceil$ multiplications and 1 addition. So there are approximately $2 \cdot log_T q$ multiplications and 2 additions in the relinearization.

  Here T is the base used in relinearization.

- Relinearization version 2:
  - Relinearization key: Only 2 elements in $R_{p \cdot q}$ are used, where p is the parameter for the relinearization. So the relinearization key is $4d \log p \cdot q$ bits.
  - Number of operations: The formula shown in Section 3.6.3.2 shows that $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ each do one multiplication, one division, and one rounding . So there are 2 multiplications, 2 divisions, and 2 roundings.

Here we use the fact that λ is the security parameter, and that elements in R have degree at most d.

3.7 Comparison of Fully Homomorphic Encryption Schemes

In this section we will give a comparison of the latest fully homomorphic encryption schemes discussed in this section. Table 3.1 compares the key and ciphertext sizes for each scheme. Note that while [BGV12] has instantations for both LWE and RLWE, we only use the result of the more efficient RLWE case.

Scheme	Based on	Secret key size	Ciphertext size	Public key size
[BV11]	LWE	$n \log q$	$(n+1) \log q$	$O(n^2 \log^2 q)$
[BGV12]	LWE and RLWE	$2d \log q$	$2d \log q$	$2dn \log q$
[Bra12]	LWE	$n \log q$	$(n+1) \log q$	$O(n^2 \log^2 q)$
[FV12]	RLWE	d	$2d \log q$	$2d \log q$

Table 3.1: Comparison of key and ciphertext sizes (in bits).

From this table we can see the similarities between the schemes based on the same assumption, with respect to key and ciphertext sizes. The schemes [BGV12] and [FV12] that are based on RLWE, use less bits than [BV11] and [Bra12] that are based on LWE. The [FV12] scheme uses the least bits overall, due to the optimization of the secret key, and a much smaller public key that is not a matrix, in contrast to the other schemes.

We will now take a look at the ideas and improvements for each work. Brakersi and Vaikuntanathan's work introduced the following concepts:

1. An LWE-based scheme with security that is better studied than Gentry's scheme that used ideal lattices.
2. Relinearization as a way of keeping the ciphertext size constant, and create a somewhat homomorphic scheme.
3. Modulus switching as a way to manage the noise and remove the need for the expensive squashing step as in Gentry's scheme.

Meanwhile, Brakersi, Gentry, and Vaikuntanathan's work introduced the following concepts:

1. A general scheme with both LWE-based and RLWE-based instantiations, with the RLWE version more efficient than Brakersi and Vaikuntanathan's scheme.
2. Relinearization generalized into key switching .
3. Modulus switching that is better implemented to reduce noise without bootstrapping.
4. Better analysis on noise than in [BV11].

Brakersi's work introduced the following concepts:

1. An LWE-based scheme that has classical reduction to GapSVP. This is in contrast with previous schemes that only have quantum reduction.
2. Invariant perspective, where the noise isn't squared by multiplication but only multiplied by a fixed polynomial .
3. Key switching as in [BGV12].

Finally, Fan and Vercauteren's work introduced the following concepts:

1. A scheme that extends Brakerski's idea to the RLWE setting. This scheme is more efficient than all the other schemes discussed.
2. Two variations of relinearization to make the scheme somewhat homomorphic. The first version is a generalization of the key switching in [BGV12]. While the second one uses a method similar to modulus switching.
3. A simpler decryption circuit with simpler analysis than previous work. The simplicity comes from reducing all cases to the optimal case, where scaling is implemented by binary right shift.

The following table gives an overview of the main ideas of the FHE schemes.

Scheme	Main Ideas
[BV11]	Relinearization, modulus switching, LWE-based
[BGV12]	Key switching, better modulus switching, LWE and RLWE-based
[Bra12]	Invariant perspective, key switching, LWE-based, classical reduction to GapSVP
[FV12]	Two versions of relinearization, more efficient RLWE-based cryptosystem, simpler decryption circuit

Table 3.2: Main ideas of FHE schemes

Chapter 4

Possible Improvements

4.1 Finding a Good Upper Bound for the Expansion Factor

In Brakerski, Gentry and Vaikuntanathan's scheme [BGV12], some of the bounds use the fact that the expansion factor $\gamma_R \leq \sqrt{d}$. However, various experiments has led to the conjecture that the upper bound for the expansion factor is less than $\sqrt{d}$. If it were possible to find a tighter upper bound for the expansion factor, it will lead to improvements in all bounds that are related to the expansion factor, such as the lemmas about modulus switching in [BGV12]. This will be a topic for future work.

4.2 Batching

The idea of batching is to compute many functions in parallel by only evaluating a single function with a larger modulus, using the idea of the Chinese Remainder Theorem. In the previous schemes based on RLWE, we used $R = \mathbb{Z}[x]/(x^d + 1)$ plaintext space of R_2. However, there is an alternative which is to use R_p which is isomorphic to $R_{p_1} \times \cdots \times R_{p_d}$ [BGV12]. Here, evaluating a function over R_p using the input $m \in R_p^n$ will evaluate the same function over R_{p_i} using the input $[m]_{p_i} \in R_{p_i}^n$ for $i = 1, \cdots, d$.

The advantage of using batching is that one can encrypt d sets of plaintext at once and perform d simultaneous evaluations on them, at the cost of one evaluation modulo p. The result will be a value that can be decomposed into d tuples, each containing the intended output. The one restriction is that all these d evaluations perform the same function, and take the same number of parameters.

Chapter 5

Applications

Some possible applications of fully homomorphic encryption are as follows.

5.1 Secure two-party computation [DFH12]

Fully homomorphic encryption can be used to implement a two-party computation protocol that is secure both against honest-but-curious adversaries, and against malicious adversaries. The protocol consists of one round with sublinear communication complexity, with the following basic idea:
The first party P_1 sends its encrypted input to the second party P_2, who uses the homomorphic property to compute ciphertexts that contain the output of the specified circuit when evaluated on P_1's (encrypted) input and his own private input. These ciphertexts are sent to P_1 who can decrypt and learn the result.

This idea alone is not secure in the malicious model, as P_2 can just perform a different function than the one intended. This can be solved by asking P_2 to provide a non-interactive zero-knowledge proof (NIZK) that he returns a ciphertext containing the correct result (without giving away any additional information). To keep the solution communication-efficient, the NIZK can be based on fully homomorphic encryption: the prover sends an encryption of his proof, the verifier then computes, using homomorphic evaluation, a ciphertext containing a bit that is 1 if and only if the proof is correct. Finally the prover gives a standard NIZK that the corresponding ciphertext indeed contains the bit 1.

5.2 Oblivious databases [LNV11, BV11]

Suppose we want to store data in the cloud. The owner of the data would like to have privacy, so that all his input data, and outputs of all operations on that data will remain secret. This can be done using fully homomorphic encryption. Users can store private data in the cloud as ciphertexts $c_1, \cdots, c_n$, all encrypted using their own public key. Then all operations or aggregate functions $f(c_1, \cdots, c_n)$ on them are done as homomorphic evaluations, using the techniques described in this text, without leaking anything to the

server. The output is sent back to the user, who can get his required result by using the decryption function on this output.

Efficient private information retrieval can also be implemented, where the user can retrieve a stored value c_i from the server without the server knowing anything about the value i. One such implementation, which requires a somewhat homomorphic and symmetric encryption, can be seen in [BV11].

Chapter 6

Conclusion

We have seen the significance of having an efficient fully homomorphic encryption scheme. We have also analyzed several fully homomorphic encryption schemes, which have some common elements:

1. An efficient lattice-based cryptosystem, with security based on the hardness of well-known lattice problems.
2. An evaluation function with definitions for c_{add} and c_{mult}, such that the noise does not rapidly increase.
3. Techniques to make the scheme fully homomorphic with this evaluation function.

We started with Gentry's scheme, the first fully homomorphic encryption scheme, and techniques such as the use of bootstrapping that have helped the rapid development of similar schemes. Gentry also provided the blueprint to construct fully homomorphic schemes:

1. Construct an encryption scheme that is somewhat homomorphic.
2. Simplify the decryption function as much as possible by the squashing technique.
3. Do bootstrapping, which is to evaluate the resulting decryption function homomorphically using the evaluation function.

Brakerski and Vaikuntanathan's scheme built on this by introducing a cryptosystem based on learning with errors (LWE), and using new techniques relinearization and modulus switching. Brakerski, Gentry and Vaikuntanathan's scheme further improved on this by using a cryptosystem based on ring learning with errors (RLWE), and modifying the techniques by Brakerski and Vaikuntanathan. This is done by generalizing the relinearization technique into something called key switching, and using modulus switching in such a way that bootstrapping is not required. Brakerski's scheme, based on LWE, used an invariant perspective where modulus switching is not required, but the secret keys used for key switching is larger than in Brakerski, Gentry and Vaikuntanathan's scheme. Finally, we saw Fan and Vercauteren's scheme which used the ideas from Brakerski's

scheme but used the LPR encryption scheme based on RLWE, which proved to be more efficient than Brakerski's scheme.

Fully homomorphic encryption has many applications, and we have discussed some of them that relate to other problems in cryptography. It is still far from practical, but there are many paths that have not been fully explored in making improvements to the existing schemes.

References

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. ACM Press. [4, 9, 10, 16, 25, 26, 28, 29, 31, 32, 36, 38, 39, 40, 41]

[Bon98] Dan Boneh. The Decision Diffie-Hellman Problem. In Joe P. Buhler, editor, *ANTS-III*, volume 1423, pages 48–63, Portland, Orgeon, USA, June, 21–25 1998. [20]

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. Technical Report 2012/078, February 19, 2012. Available at http://eprint.iacr.org/2012/078. [4, 9, 29, 30, 31, 32, 38, 40]

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In Ostrovsky [Ost11], pages 97–106. [3, 4, 9, 22, 23, 25, 26, 27, 28, 31, 38, 39, 40, 42, 43]

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure Two-Party Computation with Low Communication. In Ronald Cramer, editor, *TCC 2012*, volume 7194, pages 54–74, Taormina, Sicily, Italy, March, 19–22 2012. [4, 42]

[FV12] Junfeng Fan and Federik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Technical Report 2012/144, March 17, 2012. Available at http://eprint.iacr.org/2012/144, last version retrieved from March 22, 2012. [4, 9, 10, 32, 33, 34, 35, 37, 38, 40]

[Gam84] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO 1984*, volume 196, pages 10–18, Santa Barbara, California, USA, August, 19–22 1984. [8, 20]

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, September 2009. [3, 8, 22]

[LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can Homomorphic Encryption be Practical? In Christian Cachin and Thomas Ristenpart, editors, *ACM CCSW 2007*, pages 113–124, Chicago, Illinois, USA, October21, 2011. ACM Press. [4, 42]

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110, pages 1–23, Nice, France, May 30–June 3, 2010. Springer, Heidelberg. [17, 32]

[MR08] Daniele Micciancio and Oded Regev. Lattice-based Cryptography, 2008. [15, 16]

[Ost11] Rafail Ostrovsky, editor. *FOCS 2011*, Palm Springs, CA, USA, October, 22–25 2011. IEEE Computer Society Press. [46, 47]

[Pai02] Pascal Paillier. Composite-Residuosity Based Cryptography: An Overview. *RSA Cryptobytes*, 5(1):20–26, 2002. [8, 19]

[Reg05] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC 2005*, pages 84–93, Baltimore, MD, USA, May 22–24 2005. ACM Press. [9, 16, 29]

[Vai11] Vinod Vaikuntanathan. Computing Blindfolded: New Developments in Fully Homomorphic Encryption. In Ostrovsky [Ost11], pages 5–16. [23]