DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2023

Homomorphic Encryption

KTH Thesis Report

Daniel Sanaee

KTH ROYAL INSTITUTE OF TECHNOLOGY
MATHEMATICS

Authors

Daniel Sanaee <dsanaee@kth.se>
Mathematics
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner and supervisor

Johan Håstad
KTH Royal Institute of Technology

Abstract

The problem of constructing a secure encryption scheme that allows for computation on encrypted data was an open problem for more than 30 years. In 2009, Craig Gentry solved the problem, constructing the first fully homomorphic encryption (FHE) scheme. The challenge of constructing a homomorphic encryption scheme can be divided into three main components: 1) Constructing a decryption algorithm that is a ring homomorphism. 2) Proving CPA security. 3) Managing noise growth of evaluated ciphertexts. This thesis presents a formal mathematical background to FHE and discusses these three components. To compute on ciphertexts, the decryption algorithm needs to be homomorpic with respect to multiplication and addition operation. This means, theoretically, computing and then decrypting is equivalent to decrypting and then computing. Security is proved by reductions and hardness assumptions. The standard hardness assumptions used today is the LWE and RLWE assumptions. All existing encryption schemes are based on introducing noise when encrypting. This noise grows with each ciphertext operation and without controlling noise, decryption fails given sufficiently many operations. Gentry showed that a scheme that can evaluate its own decryption circuit can be used to reduce the noise and bootstrapped into a fully homomorphic encryption scheme.

Abstrakt

Problemet med att konstruera ett säkert krypteringssystem som tillåter beräkning på krypterad data var ett öppet problem i över 30 år. År 2009 löste Craig Gentry problemet genom att konstruera det första fullt homomorfa krypteringssystemet (FHE). Utmaningen med att konstruera ett homomorft krypteringssystem kan delas upp i tre huvudsakliga komponenter: 1) Konstruktion av en dekrypteringsalgoritm som är en ringhomomorfism. 2) Bevisning av CPA-säkerhet. 3) Hantering av brusökning i utvärderade chiffertexter. Denna uppsats presenterar en formell matematisk bakgrund till FHE och diskuterar dessa tre komponenter. För att göra beräkningar på chiffertexter måste dekrypteringsalgoritmen vara homomorf med avseende på multiplikation och addition. Detta betyder att det, teoretiskt sett, är ekvivalent att beräkna och sedan dekryptera som med att dekryptera och sedan beräkna. Säkerheten bevisas genom reduktioner och svårhetsantaganden. De standardmässiga svårhetsantaganden som används idag är LWE och RLWE-antagandena. Alla befintliga krypteringssystem är baserade på att införa brus vid kryptering. Detta brus ökar vid varje chiffertext operation och i avsaknad av kontroll över bruset kommer dekryptering misslyckas efter tillräckligt många operationer. Gentry visade att ett krypteringssystem som kan utvärdera sin egen dekrypteringskrets kan användas för att minska bruset och omvandlas till ett fullt homomorft krypteringssystem.

Acknowledgements

I want to thank my supervisor Johan Håstad for his guidance and support throughout this project. I also want to thank my family and friends for their support and encouragement.

Contents

1. Introduction

In a recent press release, Gartner estimated cloud computing to reach a market size of about 600 billion dollars in the year of 2023 [Gar23]. It is clear that the need for delegated storage and delegated computing has grown steadily as data has become more valuable. From a security perspective, delegated computation is a challenging problem. Normally, the client encrypts the data on their secure local machine and sends it to the server. The server decrypts and performs computations on the raw, unencrypted data, then re-encrypts the result and sends it back to the client, who decrypts it. For this protocol to work, the server needs access to the secret key for decrypting the data, meaning having access to the raw data. However, this raises privacy concerns. What if the server is malicious and leaks the data for profit? What if the server is compromised? To protect the client, the computation should ideally not require any information about the raw data. More generally, is there a safe way to allow for any third party to process sensitive data? How does one compute on encrypted data without first decrypting it?

Homomorphic encryption is an encryption method that allows for computation on encrypted data. To compute on ciphertexts, homomorphic encryption schemes need to satisfy what is called a homomorphic property; an operation on ciphertexts in the encrypted domain corresponds to an operation on the plaintexts in the unencrypted domain. Some encryption schemes have one natural homomorphic property, such as the multiplication operation in the RSA cryptosystem. However, to allow for general computation on ciphertexts, the scheme in question need to satisfy two types of operations; multiplication and addition of ciphertexts. It turns out that any computable function can be represented as a combination of these two operations. The question of secure, arbitrary computation on encrypted data therefore boils down to the following quesiton; does there exist a secure encryption scheme that satisfies an arbitrary (possibly infinite) combination of addition and multiplication operations on ciphertexts?

A scheme that satisfies the above is called a fully homomorphic encryption scheme. The problem of finding one has been dubbed as the ”holy grail of cryptography” and remained an open problem for more than 30 years. In 2009, Craig Gentry [Gen09], solved this problem. Gentry showed the existence of a secure encryption scheme that satisfies arbitrary multiplication and addition of ciphertext. His construction was complicated and inefficient, but it launched the development of more efficient schemes. Many research grants have been provided towards the development of more efficient schemes. The applications areas of homomorphic encryption essentially includes everything that processes sensitive information. Some examples of areas where computation on encrypted data is particularly relevant are healthcare records, machine learning, genome sequencing, navigation, electronic voting and financial records.

2. Computation theory

In this section, it is assumed that the reader has prior knowledge of deterministic- and probabilistic Turing machines as a model of computation (an excellent introduction can be found in [Gol01]). We introduce an alternative model of computation based on sets of circuits for the purpose of protection against stronger adversaries. We include the basics needed to understand cryptography and homomorphic encryption in particular.

2.1 Digital logic

Definition 1 (Circuit)**.** For $n, m \in \mathbb{N}$ and any field $(\mathbb{F}, +, \times)$, an arithmetic circuit is a vector-valued polynomial function $C\colon \mathbb{F}^n \to \mathbb{F}^m$.

A circuit C is represented by a finite directed acyclic graph with n source nodes (the n inputs) and m sink nodes (the m outputs). The internal nodes of the circuit are called *gates* and is stacked in layers. For more details about the structure of a circuit, see [Gol08] or [MF21]. The number of nodes in C is called its *size* and is denoted $|C|$. The longest path in C is called its *depth*. A circuit is called *Boolean* when the field is $\mathbb{F}_2$ and each gate takes at most 2 inputs. Boolean circuits and arithmetic circuits are equivalent in the sense that the set of functions that can be computed by an arithmetic circuit is equal to the set of functions that can be computed by a Boolean circuit.[1] If the input is a string of bits, we assume the circuit is Boolean.

We consider circuits as algorithms and use them as an alternative approach to the traditional Turing machine model of computation.[2] Notice that any given circuit, C, can only compute on inputs of the same length whereas a Turing machine M takes inputs of any size n. However, a circuit always halts on a given input whereas a Turing machine may not. For the purpose of our discussion relating to cryptography, we assume every Turing machines halts unless otherwise stated. To allow circuits to handle arbitrary length inputs we consider families of circuits.

[1]A Boolean circuit is an arithmetic circuit. Conversely, any arithmetic circuit can be simulated by representing the inputs and outputs as a bitstring and utilizing that XOR and AND is a complete set of gates.

[2]The reason for this alternative model is to assume adversaries are computationally ”stronger”. See Theorem 1.

Definition 2 (Circuit family [MF21])**.** A circuit family $C = \{C_n\}_{n \in \mathbb{N}}$ is an indexed set of circuits $C_n \colon \mathbb{F}^{n+r} \to \mathbb{F}^m$ where $r, m = \text{poly}(n)$.

For any input x with length n, $C(x) \overset{\text{def}}{=} C_n(x)$. For each circuit $C_n \in C$, r represents the random coins used. If $r = 0$ for all n then C is a deterministic circuit family. A circuit family is said to have polynomial-size if there exists a polynomial p such that $|C_n| < p(n)$ for all n.

2.2 Complexity classes

Definition 3 (Complexity Class P)**.** P is the set of languages $\mathscr{L}$ such that there exists a deterministic polynomial-time Turing machine M satisfying $M(x) = 1 \iff x \in \mathscr{L}$

Definition 4 (Complexity Class BPP)**.** BPP is the set of languages $\mathscr{L}$ such that there exists a probabilistic polynomial-time (PPT) Turing machine M satisfying

$$\Pr[M(x) = 1] \geq 2/3 \text{ if } x \in L$$
$$\Pr[M(x) = 1] \leq 1/3 \text{ if } x \notin L$$

Definition 5 (Complexity Class P/poly)**.** P/poly is the set of languages $\mathscr{L}$ such that there exists a polynomial-size circuit family C satisfying $C(x) = 1 \iff x \in \mathscr{L}$

Informally speaking, circuit families is a stronger model of computation than the PPT Turing machine model in the sense that if there exists a PPT Turing machine for deciding a problem, then there also exists a circuit family that can decide the same problem. The formal statement is as follows:

Theorem 1. P $\subseteq$ BPP $\subsetneq$ P/poly

The first inclusion follows from the fact that if there exists a deterministic Turing machine that decides a language, then that same machine can be seen as a PPT machine that ignores a given input sequence of coin tosses. For the second inclusion, consider a language $\mathscr{L} \in$ BPP and a corresponding PPT machine M for $\mathscr{L}$. Then, for any given input x_n with length n, at least $2/3$ of the set of all possible coin toss sequences are good (good r means $M(x_n; r) = 1 \iff x_n \in \mathscr{L}$). This means that there exists at least 1 sequence of coin tosses that yields the correct result for $2/3$ of the possible inputs of length n. Consider machine M' that on input x_n runs $M(x_n)$ many (still $\text{poly}(n)$) times and outputs the majority result. Then, the error probability vanishes exponentially, meaning there are exponentially few coin toss sequences that are bad for M' (see [AB09; Gol01] for more detail). Therefore there exist a coin toss sequence, r_n, that yields the correct result for all inputs of length n. Consider circuit $C_n \colon \{0,1\}^{n+|r_n|} \to \{0,1\}$ with r_n hardcoded as inputs where C_n simulates M' using r_n, i.e., $C_n(x_n) = M'(x;r)$. Therefore $C_n(x) = 1 \iff x \in \mathscr{L}$ and C decides $\mathscr{L}$.

Interestingly the first inclusion is speculated to be set equivalence [AB09, pp. 126], meaning that a deterministic machine could decide the same languages as a probabilistic one. The second inclusion is proper since every unary language is in P/poly whereas undecidable ones are not in BPP (see [AB09, pp. 110] for details). In this sense, circuit families are a stronger model of computation than PPT Turing machines. We capture this notion with uniformity.

Definition 6 (Uniform circuit family)**.** A circuit family $\{C_n\}_{n\in\mathbb{N}}$ is uniform if there exists a polynomial-time Turing machine M such that $M(1^n)$ outputs the description of C_n for all $n \in \mathbb{N}$.

A uniform circuit family is polynomial size. The converse is not necessarily true. A family that is not uniform is said to be a non-uniform circuit family. Note that Turing machines are at least as strong as uniform circuit families. More formally, if a uniform circuit family decides $\mathscr{L}$ then there exists a polynomial-time Turing machine that decides $\mathscr{L}$.[3] Simply construct the polynomial-time Turing machine that given any input $x \in \mathscr{L}$, first generates a description of $C_{|x|}$ and then simulates $C_{|x|}(x)$. In other words, the non-uniform circuit families are stronger than the polynomial-time Turing machines.

[3]The converse is also true, meaning deterministic polynomial-time Turing machines are exactly as powerful as uniform circuit families. See [AB09, pp. 111] for details

3. Mathematical foundations

3.1 Probability theory

Homomorphic encryption schemes are based on noise. As we will see in Subsection 5.1, noise can be sampled from discrete spaces in accordance with a distribution. Since distributions are central in cryptography, it is important they are understood. A distribution is a probability measure on a measurable space $(S, \mathcal{S})$. Typically, probability distributions are associated with random variables; however, in the absence of random variables, distributions are understood as a specified measure function on the given measurable space (See [Kal21, pp. 83]).

Definition 7 (Discrete distribution measure of a random variable)**.** Consider a probability space $(\Omega, \mathcal{F}, \Pr)$ and a discrete measurable space $(S, \mathcal{S})$. Let $X \colon \Omega \mapsto S$ be a random variable. The discrete distribution measure of X, or simply *distribution* of X, χ is defined as follows

$$\begin{aligned} \chi \colon \mathcal{S} &\to [0,1] \\ \{x\} &\mapsto \Pr[X = x] \end{aligned}$$

We say that χ is the distribution or *law* of X, denoted $\mathcal{L}(X)$.

Remark. Since χ is a measure on a discrete space, the description of the singletons are sufficient. More explicitly, $\chi(A) = \sum_{x \in A} \Pr[X = x]$ for $A \in \mathcal{S}$

A distribution measure for a random variable is a probability measure on the sample space $(S, \mathcal{S})$, as opposed to on the outcome space $(\Omega, \mathcal{F})$. For a given probability space, any random variable X defined on it has a distribution associated with it. We write $x \leftarrow X$ or $x \leftarrow \chi$ for sampling the outcome x from X assuming χ is its distribution. When X is a space we mean that x is uniformly sampled from X.

Definition 8 (Ensembles)**.** Let I be a countable index set. A *probability ensemble* indexed by I (or just ensemble) is a sequence of random variables $(X_i)_{i \in I}$. A *distribution ensemble* indexed by I is a sequence of distributions $(\chi_i)_{i \in I}$

In this paper we will exclusively use $\mathbb{N}$ as the index set. For example, the encryption function is a random variable (PPT algorithm), meaning that a single message m corresponds to many valid ciphertexts. By varying the security parameter of the scheme we construct the probability ensemble $\{Enc(pk, m)\}_{pk \in \mathbb{N}}$

Definition 9 (Statistical distance)**.** Let S_n be finite set for all $n \in \mathbb{N}$ and let $X = (X_n \colon \Omega_n \to S_n)_{n \in \mathbb{N}}$, $Y = (Y_n \colon \Omega_n \to S_n)_{n \in \mathbb{N}}$ be two ensembles. The *statistical distance* is defined as

$$\Delta_{X,Y}(n) \overset{\text{def}}{=} \frac{1}{2} \sum_{\alpha \in S_n} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|.$$

Let $\chi = \{\chi_n\}_{n \in \mathbb{N}}$, $\psi = \{\psi_n\}_{n \in \mathbb{N}}$ be two discrete distribution ensembles on the same measurable spaces $\{(S_n, \mathcal{S}_n)\}_{n \in \mathbb{N}}$. The statistical distance is defined as

$$\Delta_{\chi,\psi}(n) \overset{\text{def}}{=} \frac{1}{2} \sum_{\alpha \in S_n} |\chi_n(\{\alpha\}) - \psi_n(\{\alpha\})|$$

Remark. This definition for distribution ensembles assumes every singleton is measurable. A more general definition is $\Delta(\chi, \psi) \overset{\text{def}}{=} \sup_{A \in \mathcal{S}} |\chi(A) - \psi(A)|$

Definition 10 (Negligible function)**.** Let $f \colon \mathbb{N} \to [0, 1]$. f is negligible if for all positive polynomials $p(\cdot)$, there exists an N such that for all $n > N$, $f(n) < \frac{1}{p(n)}$. We say $f = \text{negl}(n)$.

Definition 11 (Perfectly indistinguishable)**.** Two ensembles (probability or distribution) X and Y are *perfectly indistinguishable* if $\Delta_{X,Y}(n) = 0$ for all n.

Definition 12 (Statistically indistinguishable)**.** Two ensembles (probability or distribution) X and Y are *statistically indistinguishable* if $\Delta_{X,Y}(n) = \text{negl}(n)$.

A desirable property of encryption schemes is to make it unfeasible for adversaries to distinguish encryptions. For two random variables (or distributions) X and Y, we want every adversary C to not be able to distinguish samples from X and Y. More precisely, an adversary tasked with identifying whether given samples are from X (C outputs 0) or from Y (C outputs 1) should have roughly the same probability of success irrespective of which the samples are generated from. The formal definition is as follows:

Definition 13 (Computationally indistinguishable [Gol01])**.** Two ensembles (or distribution ensembles) X and Y are computationally indistinguishable if, for every polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$,

$$\mathrm{Adv}_{X,Y}(n) \stackrel{\mathrm{def}}{=} |\Pr[C(X_n) = 1] - \Pr[C(Y_n) = 1]| = \mathrm{negl}(n).$$

$C(X_n), C(Y_n)$ means that the adversary has access to random oracle returning a sample from X_n and Y_n respectively. The point is that, for every adversary C guessing the samples are from one of the random variables or distributions (1 in this case represents from Y_n), the probability of being correct is, up to negligible error, equal to the probability of being incorrect. For example, an adversary always guessing 1 has 100% probability of being incorrect when given samples from X_n and 100% probability of being correct when given samples from Y_n. The intuition is that the sample does not make a noticable difference on the guess and that the advantage in distinguishing samples goes to zero fast (faster than the inverse of all polynomials).

Perfect distinguishability implies statistical indistinguishability as 0 is negligible. Statistical indistinguishability implies computational indistinguishability. To see why, assume $\Delta_{X,Y}(n) = \mathrm{negl}(n)$ and consider any circuit family C. Since X_n and Y_n are defined on the same space we have

$$\begin{aligned}\mathrm{Adv}_{X,Y}(n) &= |\Pr[C(X_n) = 1] - \Pr[C(Y_n) = 1]| \\ &\leq \sum_{\alpha\in S_n} |\Pr[C(X_n = \alpha) = 1 \mid X_n = \alpha]\Pr[X_n = \alpha] \\ &\qquad - \Pr[C(Y_n = \alpha) = 1 \mid Y_n = \alpha]\Pr[Y_n = \alpha]| \\ &\leq \sum_{\alpha\in S_n} |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]| = 2\Delta_{X,Y}(n) = \mathrm{negl}(n)\end{aligned}$$

where the first inequality is due to the triangle inequality and the second is due to conditional probability being upper bounded by 1. Intuitively, there does not exist enough distinguishable outcomes to distinguish the ensembles even if there exists an adversary that always correctly recognize an outcome from X and Y respectively.

3.2 Discrete Gaussian Distribution

To add discrete noise to ciphertexts, it is typical to use the discretized Gaussian distribution introduced by Micciancio and Regev [MR07]. Since this distribution is used so frequently, we give a brief presentation of it here. A continuous Gaussian distribution in $\mathbb{R}^n$ centered at $\vec{c}$ where each coordinate is independently sampled from normal

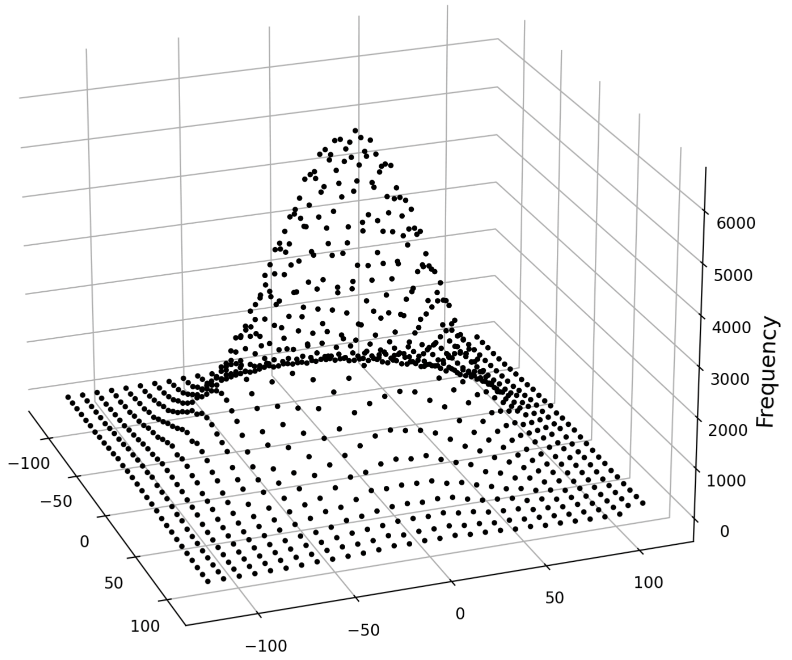

Figure 1: Discrete Gaussian distribution $\mathrm{D}_{\mathbb{Z}^2,100,\vec{0}}$ with 1 million samples. Every point has integer coordinates. See code in Appendix A.

distirbution with standard deviation σ, then the multivariate Gaussian distribution is simplified to $\rho_{\vec{c}}(\vec{x};\sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\|\vec{x}-\vec{c}\|^2\right)$. By introducing the scale factor $s \overset{\text{def}}{=} \sqrt{2\pi}\sigma$ we get $\rho_{s,\vec{c}}(\vec{x}) = \frac{1}{s^n}\exp\left(-\frac{\pi}{s^2}\|\vec{x}-\vec{c}\|^2\right)$. The goal is to discretize the gaussian over lattices, see Section 5, by restricting the domain to lattice points. For a given lattice $L \subset \mathbb{R}^n$ we define the normalization constant $\rho_{s,\vec{c}}(L) \overset{\text{def}}{=} \sum_{\vec{x}\in L} \rho_{s,\vec{c}}(\vec{x})$.

Definition 14 (Discrete Gaussian distribution)**.** Let $L \subset \mathbb{R}^n$ be a lattice. Then the discrete gaussian distribution is defined as

$$\forall \vec{x} \in L\colon \quad \mathrm{D}_{L,s,\vec{c}}(\vec{x}) \overset{\text{def}}{=} \frac{\rho_{s,\vec{c}}(\vec{x})}{\rho_{s,\vec{c}}(L)}.$$

Definition 15 (β-bounded distribution)**.** Let χ be a distribution over the integers. χ is β-bounded if $x \leftarrow \chi \implies |x| < \beta$

Remark. It is also possible (and common) to define a β-bounded distribution such that $|x| \geq \beta$ with negligible probability.

Subgaussian distributions

In this section we introduce subgaussian distributions. We use subgaussian distributions for a tighter error analysis when discussing the GSW scheme in Chapter 8. Much of this section can be found in [AP13].

Definition 16 (Subgaussian distribution)**.** A distribution χ over $\mathbb{R}$ is subgaussian with parameter $K > 0$ (K not nesessarily constant) if for every non negative $t \in \mathbb{R}$.

$$\chi(\{x \in \mathbb{R} \text{ such that } |x| > t\}) \leq 2e^{-\pi\frac{t^2}{K^2}}$$

Or equivalently, a real valued random variable X has subgaussian distribution if

$$\Pr[|X| > t] \leq 2e^{-\pi\frac{t^2}{K^2}}$$

We say X is a subgaussian random variable.

The intuition of subgaussian distributions is that the probability of a random variable X being taking a large value is less than the probability of a Gaussian taking the same value. In other words, the tails of a subgaussian distribution decays at least as fast as the tails of a Gaussian distribution. In particular, the β-bounded distributions are subgaussian with parameter $\beta\sqrt{2\pi}$. To see why, clearly, for all $t \geq \beta$, $\Pr[|X| > t] = 0$ whereas the exponential function is non-negative. Furthermore, for $t < \beta$ we have $2e^{-\pi\frac{t^2}{2\pi\beta^2}} > 2e^{-\pi\frac{\beta^2}{2\pi\beta^2}} = 2e^{-\frac{1}{2}} > 1 > \Pr[|X| > t]$. Hence, the definition is satisfied for all non-negative t.

Subgaussian variables are closed under scalar multiplication in the sense that if X is a subgaussian random variable with parameter K, then for any non-zero real number a, aX is subgaussian with parameter $|a|K$. This follows from $\Pr[|aX| > t] = \Pr[|X| > \frac{t}{|a|}] \leq 2e^{-\pi\frac{t^2}{(|a|K)^2}}$. Furthermore, it can be shown by using moment generating functions that a finite sum of n subgaussian random variables with parameters $K_1, \ldots, K_n$ is subgaussian with parameter $\sqrt{\sum_{i=1}^{n} K_i^2}$.

We extend the definition of subgaussian random variables to vectors of subgaussian random variables. Let $u \in \mathbb{R}^n$ be any fixed unit vector and let $\vec{X} = (X_1, \ldots, X_n)$ be a vector of random variables. We say $\vec{X}$ is subgaussian with parameter K if $\left\langle u, \vec{X} \right\rangle$ is subgaussian with parameter K. The intuition is that for any direction, the probability of sampling points far away from origin decays at least as fast as a gaussian. Our focus is on random vectors of independent subgaussian entries. In this case, the vector is subgaussian with the same parameter as the maximum parameter of the entries (see Proposition 5.10 in [Ver11]). In other words, let $\vec{X} = (X_1, \ldots, X_n)$ be a vector of independent subgaussian random variables with parameters $K_1, \dots, K_n$. Then $\vec{X}$ is subgaussian with parameter $\max_{i \in [n]} K_i$.

By considering each element individually, it is clear that multiplication with non-negative scalar and addition of vectors preserves subgaussianity in the same way. An important application is the inner product of a vector with a subgaussian vector. Let $\vec{e} = (e_1, \dots, e_n)$ be a vector of independent entries where each entry is sampled from a β-bounded distribution (i.e., $\vec{e}$ is subgaussian). Let $\vec{X} = (X_1, \dots, X_n)$ be a vector of independent subgaussian random variables with parameters $K_1, \dots, K_n$ for some fixed constants $K_1, \dots, K_n$. Denoting $K = \max_{i \in [n]} K_i$ then gives $\left\langle \vec{e}, \vec{X} \right\rangle$ is subgaussian with parameter $\sqrt{\sum_{i=1}^n e_i^2 K_i^2} \leq K\sqrt{\sum_{i=1}^n e_i^2} = K\|\vec{e}\| = O(\|\vec{e}\|)$. As a last point, we introduce a bound on the euclidean norm of a subgaussian vector.

Theorem 2 (Lemma 2.1 [AP13])**.** *Let $\vec{X}$ be an n dimensional subgaussian vector with parameter K, containing mutually independent entries. Then there exists real positive constant C such that*

$$\Pr[\|\vec{X}\| > CK\sqrt{n}] \leq 2^{-\Omega(n)}$$

In particular, the length $\|\vec{X}\|$ is $O(K\sqrt{n})$ except with negligible probability.

3.3 Embedding integers as permutations

In this section we show how to embed integers in a finite group as a tuple of cyclic shifts by showing $\mathbb{Z}_q \cong S_{r_1} \times \cdots \times S_{r_t}$ where $q = \prod_{i=1}^t r_i$ for pairwise coprime r_i. Representing integers in this way is nesessary for the error analysis in the implementation of the GSW scheme, see Chapter 8. The isomorphism is established through the Chinese Remainder Theorem and Cayley's theorem.

Theorem 3 (Chinese Remainder Theorem)**.** *Let $r_1, \dots, r_t$ be pairwise coprime integers and $q = \prod_{i=1}^t r_i$. The system of congruences*

$$\begin{aligned} x &\equiv a_1 \pmod{r_1} \\ &\vdots \\ x &\equiv a_t \pmod{r_t} \end{aligned}$$

has a unique solution $x \in \mathbb{Z}_q$. We say $(a_1, \dots, a_t)$ is the RNS representation of x with respect to the moduli set $\{r_1, \dots, r_t\}$.

The Chinese Remainder Theorem allows us to construct an isomorphism from $\mathbb{Z}_q$ to $\mathbb{Z}_{r_1} \times \cdots \times \mathbb{Z}_{r_t}$.

Definition 17 (CRT isomorphism)**.** We define the *CRT isomorphism* $\mathbb{Z}_q \to \mathbb{Z}_{r_1} \times \cdots \times \mathbb{Z}_{r_t}$ as a mapping of x to its RNS representation $(a_1, \ldots, a_t)$ with respect to the moduli set $\{r_1, \ldots, r_t\}$.

Remark. To compute the CRT isomorphism on a given input x we simply compute $x \bmod r_i$ for each i (we will not need the inverse mapping but it can be computed using the Extended Euclidean Algorithm).

We will now discuss how to embed the finite additive group $\mathbb{Z}_r$ into the symmetric group S_r. The symmetric group S_r is the group of all permutations of the set $\{1, \ldots, q\}$. A standard way to represent a permutation $\pi\colon \{(x_1, \ldots, x_r) \mid x_a \neq x_b,\ a, b \in [r]\} \to \{(x_1, \ldots, x_r) \mid x_a \neq x_b,\ a, b \in [r]\}$ in S_r is to list where each entry is mapped to; $(x_1, \ldots, x_r) \overset{\pi}{\mapsto} (\pi(x_1), \ldots, \pi(x_r))$. In this paper we will use a binary representation by using a square permutation matrix $M^\pi \in \{0,1\}^{r \times r}$ such that $M^\pi_{i,j} = 1$ if and only if π maps the j:th entry to the i:th entry. In other words, column j represents the mapping of element j. Since every entry is mapped to exactly one other entry, every column has exactly 1 non-zero entry. Since a permutations is a bijection, surjectivity implies every row has a non-zero entry. Therefore, every row and every column has exactly 1 non-zero element. M^π contains exactly r entries equal to 1 and $r^2 - r$ entries equal to 0. For example, consider a cyclic shift permutation $\pi \in S_3$ defined as $(\pi(x_1), \pi(x_2), \pi(x_3)) = (x_3, x_1, x_2)$. Then the corresponding permutation matrix M^π is

$$M^\pi = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

A particular permutation group of interest is the cyclic shift group C_r. C_r is the subgroup of the symmetric group S_r consisting of permutations shifting the input sequence by a fixed number of positions. Every element of this group can be represented by only the first column of the permutation matrix since the shift of the first element in the sequence must be the same as the shift for every other element. In other words, every element of C_r can be represented by a r dimensional indicator vector $\pi \in \{0,1\}^r$. Our above example of a cyclic shift of 1 position to the right can therefore be represented by the indicator vector $\pi = (0,1,0)^T$.

Theorem 4 (Cayley's theorem)**.** *Every group G is isomorphic to a subgroup of the symmetric group $S_{|G|}$.*

In particular, the additive group $\mathbb{Z}_r$ is isomorphic to C_r through the following natural embedding isomorphism.

Definition 18 (Embedding isomorphism)**.** We define the *embedding isomorphism* $\mathbb{Z}_r \to C_r$ as a mapping of x to the indicator vector $(b_0, \ldots, b_x, \ldots, b_{r-1})$ where $b_i = 1$ if and only if $i = x$.

By using the CRT isomorphism and the embedding isomorphism we can represent an element in $\mathbb{Z}_q$ as a direct sum of cyclic shifts. Consider the following example showing how to represent an element in $\mathbb{Z}_q$ into $S_{r_1} \times \cdots \times S_{r_t}$. Say we want to represent the integer $7 \in \mathbb{Z}_{15}$ as a direct sum of cyclic shifts. I.e., $q = 15 = 3 \times 5$, $t = 2$, $r_1 = 3$ and $r_2 = 5$. We have $x \bmod 3 = 1$ and $x \bmod 5 = 2$. The RNS representation of 7 is therefore $(1, 2)$ with respect to the moduli set $\{3, 5\}$. We can now represent x in $S_3 \times S_5$ by using the embedding isomorphism on each element of the RNS representation. We have $(1, 2) \mapsto ((0, 1, 0)^T, (0, 0, 1, 0, 0)^T)$. Thus, the element 7 in $\mathbb{Z}_{15}$ is represented by a tuple of two cyclic shifts, $((0, 1, 0)^T, (0, 0, 1, 0, 0)^T) \in S_3 \times S_5$.

3.3.1 Efficient representation of integers as cyclic shifts

As it stands, representing integers as cyclic shifts can be innefficient. For instance, for a prime q, an element in $\mathbb{Z}_q$ is represented in binary using $\log q$ bits whereas the corresponding cyclic shift is represented with an indicator vector using q bits, thus yielding an exponential expansion in the representation size. For cyclic shift representation to be efficient, we want to choose modulus q yielding pseudo-constant bit size representation. In other words, the goal is to guarantee choise of q such that representing an integer using a tuple of cyclic shifts requires $O(\text{polylog(q)})$ bits. To do so, we want to find $q = \prod_{i=1}^{t} r_i$ such that $\max\limits_{i \in [t]} r_i = O(\log q)$ and $t = O(\frac{\log q}{\log \log q})$. This would yield that every $x \in S_{r_1} \times \cdots \times S_{r_t}$ can be represented in $O(\frac{\log^2 q}{\log \log q}) = \tilde{O}(1)$. We will now show that such a modulus q exists and how to find it.

Definition 19. The maximal prime powers bounded by an positive integer r is defined as the set

$$\text{MPP}(r) \stackrel{\text{def}}{=} \{p^{\lfloor \log_p r \rfloor} \,|\, p \leq r, \text{ p is prime}\}.$$

An element in MPP(r) is called a maximal prime power.

For example, the maximal prime powers bounded by 10 is $\text{MPP}(10) = \{2^3, 3^2, 5^1, 7^1\} = \{8, 9, 5, 7\}$.

Theorem 5 (Lemma 2.2 [AP13]). *Let $r \geq 7$ be an integer. Then*

$$\prod_{r_i \in MPP(r)} r_i \geq e^{\frac{3r}{4}}.$$

The idea is to choose q as the product of maximal prime powers bounded by some $r = O(\log q)$. In other words, we consider $t = |\text{MPP}(r)|$ and $\{r_1, \ldots, r_t\} = \text{MPP}(r)$. We can efficiently find such q by first determining an acceptable lower bound $q_0 \leq q$. Since $r \geq 7$, we require $q_0 \geq e^{\frac{3\times 7}{4}} > 190$. The second step is to find an r such that the product of its maximal prime powers is greater than or equal to our lower bound q_0. Choosing $r = \lceil \frac{4}{3} \log q_0 \rceil$ works since $q = \prod_{r_i \in \text{MPP}(r)} r_i$ is, by Theorem 5, at least $e^{\frac{3r}{4}} = e^{\frac{3}{4}\lceil \frac{4}{3} \log q_0 \rceil} \geq e^{\log q_0} = q_0$.

Note that r is logarithmic with respect to q, implying $\max_{i \in [t]} r_i = O(\log q)$. We now show that $t = O(\frac{\log q}{\log \log q})$.

Theorem 6 (Prime number theorem). *Let $f(x)$ be the number of primes less than or equal to the real value x. Then*

$$\lim_{x \to \infty} \frac{f(x)}{x / \ln(x)} = 1$$

Note that prime number theorem gives that the number of primes bounded by an integer x is $O(x/\log(x))$. In particular, $\text{MPP}(r)$ contains exactly one element per prime bounded by r, meaning $t = O(\frac{r}{\log r}) = O(\frac{\log q}{\log \log q})$.

As an example, say we want a modulus $q \geq 1000$. Then $r = \lceil \frac{4}{3} \log 1000 \rceil = 14$. We have $q = \prod_{r_i \in \text{MPP}(14)} r_i = 2^3 \times 3^2 \times 5^1 \times 7^1 \times 11^1 \times 13^1 = 360360$. We can now represent an element $x \in \mathbb{Z}_{360360}$ as a tuple of cyclic shifts in $S_8 \times S_9 \times S_5 \times S_7 \times S_{11} \times S_{13}$. Note that $t = |\text{MPP}(14)| = 6 = O(\frac{\log q}{\log \log q})$ and $\max_{i \in [t]} r_i = \max\{\text{MPP}(14)\} = 13 = O(\log q)$.

4. Cryptographic primitives

In the following definitions we let the message space be denoted $\mathcal{X}$, the ciphertext space be denoted $\mathcal{Y}$, and the key space be denoted $\mathcal{K} = \mathcal{K}_{pk} \times \mathcal{K}_{sk}$.

Definition 20 (Encryption scheme)**.** A correct asymmetric *encryption scheme* $\mathcal{E}$ = (KeyGen, Enc, Dec) is a triple of algorithms satisfying the following:

- KeyGen : $\{1\}^* \to \mathcal{K}$ is PPT given by $1^\lambda \mapsto (pk, sk)$.
- Enc : $\mathcal{K}_{pk} \times \mathcal{X} \to \mathcal{Y}$ is PPT given by $(pk, m) \mapsto c$.
- Dec : $\mathcal{K}_{sk} \times \mathcal{Y} \to \mathcal{X}$ is deterministic and satisfies $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \implies \text{Dec}(sk, \text{Enc}(pk, m)) = m$.

Remark. This paper is only concerned with encryption schemes where decryption always works (correct) and where more than one key is used (asymmetric). Thus, every encryption scheme is assumed to be a correct asymmetric encryption scheme. Furthermore, the decryption algorithm can be considered a PPT algorithm that with probability 1 outputs the correct message for the correct secret key. In other words, the algorithm ignores the input coin toss sequence.

In this paper, we consider homomorphic encryption (HE) schemes. These schemes include a fourth algorithm, Eval, called the evaluation algorithm which is used by the server calculating on encrypted data.

Definition 21 ($\mathcal{C}$-homomorphic encryption scheme)**.** An encryption scheme $\mathcal{E}$ is a $\mathcal{C}$-*homomorphic encryption scheme* for the set of circuits $\mathcal{C}$ if there exists an extra algorithm Eval such that for any $C \in \mathcal{C}$ taking t inputs the following condition holds:

- Eval: $\mathcal{K}_{pk} \times \mathcal{C} \times \mathcal{Y}^* \to \mathcal{Y}$ is PPT and satisfies $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \implies \text{Dec}(sk, \text{Eval}(pk, C, \langle \text{Enc}(pk, m_1), \ldots, \text{Enc}(pk, m_t)\rangle)) = C(m_1, \ldots, m_t)$

We say $\mathcal{E}$ can evaluate all circuits in $\mathcal{C}$ and is $\mathcal{C}$-homomorphic.

The evaluation algorithm runs the ciphertexts through the permissible circuit while also satisfying the requirement that decrypting the resulting ciphertext yields the same result

$$\begin{tikzcd}
\mathcal{Y}^t \arrow[r, "\mathrm{Eval}(pk{,}C{,}\langle c_1{,}\ldots{,}c_t\rangle) = c_f"] \arrow[d, "\mathrm{Dec}(sk{,}\langle c_1{,}\ldots{,}c_t\rangle)"'] & \mathcal{Y} \arrow[d, "\mathrm{Dec}(sk{,}c_f)"] \\
\mathcal{X}^t \arrow[r, "C(m_1{,}\ldots{,}m_t)"'] & \mathcal{X}
\end{tikzcd}$$

Figure 2: The decryption homomorphism. The path through $\mathcal{Y}$ represents computing before decrypting. The path through $\mathcal{X}^t$ represents decrypting before computing.

as the plaintexts running through the circuit. To its disposal, the evaluation algorithm is given the public key. The ciphertexts returned by the Eval algorithm are called *evaluated ciphertexts* (suggesting the circuit has evaluated the ciphertexts) and those returned by the encryption algorithm are called *fresh ciphertexts.* Remark that correctness is only guaranteed if the Eval algorithm is given fresh ciphertexts. If the circuit corresponds to the computable function f acting on a vector c of ciphertexts, we denote the evaluated ciphertexts c_f (i.e., $c_f \overset{\text{def}}{=} \mathrm{Eval}(pk, f, c)$). Similarly, for the vector m of plaintexts, we denote the evaluated plaintexts m_f (i.e., $m_f \overset{\text{def}}{=} f(m)$). Thus, the condition for the Eval algorithm can be written $(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda) \implies \mathrm{Dec}(sk, c_f) = m_f$. In a homomorphic encryption scheme that supports one addition and multiplication of fresh ciphertexts, the decryption function is a ring homomorphism. Consider a valid key pair (sk, pk) which are, for notational simplicity, hard-wired into the decryption and evaluate functions respectively, plaintext ciphertext pairs $(m_1, c_1 = \mathrm{Enc}_{pk}(m_1))$, $(m_2, c_2 = \mathrm{Enc}_{pk}(m_2))$ and circuits $C_+(m_1, m_2) \overset{\text{def}}{=} m_1 + m_2$ and $C_\times(m_1, m_2) \overset{\text{def}}{=} m_1 \times m_2$ that can be evaluated by the scheme.

$$\begin{aligned}
\mathrm{Dec}_{sk}(c_1 + c_2) &= \mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(C_+, \langle c_1, c_2\rangle)) = C_+(m_1, m_2) = m_1 + m_2 \\
&= \mathrm{Dec}_{sk}(c_1) + \mathrm{Dec}_{sk}(c_2) \\
\mathrm{Dec}_{sk}(c_1 \times c_2) &= \mathrm{Dec}_{sk}(\mathrm{Eval}_{pk}(C_\times, \langle c_1, c_2\rangle)) = C_\times(m_1, m_2) = m_1 \times m_2 \\
&= \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2)
\end{aligned}$$

The main idea behind a homomorphic encryption scheme is to give a server encrypted data so that it can compute on that data and return the answer in encrypted form. However, the definition provided allows for trivial homomorphic encryption schemes where the server does nothing. More specifically, consider any set of circuits $\mathcal{C}$ and let $\mathrm{Eval}(pk, C, \langle c_1, \ldots, c_t\rangle) = (C, \langle c_1, \ldots, c_t\rangle)$. Eval takes a description of a circuit and a tuple of ciphertexts, one for each input wire of the circuit, and simply outputs the description of the circuit together with the given tuple. Clearly, Eval runs in polynomial time. Consider a decryption algorithm that, if given an input of this form, first decrypts the t ciphertexts and then computes m_f using C. To ensure that the server actually processes the given inputs we introduce compactness.

Definition 22 (Compactness)**.** A $\mathcal{C}$-homomorphic encryption scheme is compact if there exists a polynomial $p(\lambda)$ such that for all $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$, for every $C \in \mathcal{C}$ taking any number t inputs and any $c \in \mathcal{Y}^t$, the size of the output $\text{Eval}(pk, C, \langle c_1, \dots, c_t \rangle)$ is less than $p(\lambda)$. We say that the scheme compactly evaluates $\mathcal{C}$.

For a compact $\mathcal{C}$-homomorphic encryption scheme, the size of the output is independent of the circuit function used. In particular, the previous, trivial homomorphic encryption scheme where $\text{Eval}(pk, C, \langle c_1, \dots, c_t \rangle) = (C, \langle c_1, \dots, c_t \rangle)$ is not compact for any set of circuits with unbounded circuit size, which includes circuit families with circuits that do not ignore all except for constantly many inputs, meaning essentially every application of a HE scheme.

Definition 23 (Fully Homomorphic Encryption (pure FHE))**.** Let $\mathcal{C}$ be the class of all circuits. An encryption scheme $\mathcal{E}$ is a fully homomorphic encryption (pure FHE) scheme if it is $\mathcal{C}$-homomorphic and compactly evaluates $\mathcal{C}$.

For a scheme to be fully homomorphic it is required that it can evaluate circuits of arbitrary size. Many times it suffices to consider only circuits of a beforehand specified depth, L, as any deeper circuits are irrelevant to the application. The following definition capture schemes that can evaluate any set of circuits with depths bounded by the client.

Definition 24 (Leveled fully homomorphic encryption (leveled FHE))**.** An encryption scheme $\mathcal{E}$ with the KeyGen algorithm modified is a leveled fully homomorphic encryption scheme if it satisfies the following:

- $\text{KeyGen} : \{1\}^* \times \{1\}^* \to \mathcal{K}$ is PPT given by $(1^\lambda, 1^L) \mapsto (pk, sk)$.
- Let $\mathcal{C}_L$ be the set of circuits with depth less than or equal to L. Then $\mathcal{E}$ is $\mathcal{C}_L$-homomorphic.
- $\mathcal{E}$ compactly evaluates the set of all circuits.

Remark. Notice that the length of the evaluated ciphertexts in a leveled FHE scheme is independent of the depth.

For any specified circuit C, a leveled FHE scheme can evaluate it by choosing sufficiently large depth parameter, L. For a pure FHE scheme, the circuit does not need to be specified. A pure FHE scheme can dynamically compute any circuit whereas the leveled FHE scheme requires the circuit chosen a priori.

4.1 Security definitions

In this paper, semantic security refers to security against chosen-plaintext attack (CPA). The definition relates to the following game where the challenger possess the secret key and the player is the adversary trying to break the scheme. Consider encryption scheme (KeyGen, Enc, Dec, Eval) and polynomial-size Boolean circuit family $C = \{C_n\}_{n\in\mathbb{N}}$. The CPA game is defined with the Boolean function $\text{CPA}_C(\lambda)$ as follows:

1. **Setup**: Challenger samples $pk \leftarrow \text{KeyGen}$ and sends it to player.
2. **Choose**: Player C selects two distinct plaintext messages $(m_0, m_1) \leftarrow C(pk)$ of the same length, and sends them to the challenger.
3. **Encrypt**: The challenger randomly picks a bit $b \in \{0,1\}$ and encrypts the message m_b. The encrypted message $c \overset{\text{def}}{=} \text{Enc}(pk, m_b)$, called challenge ciphertext, is sent to the player.
4. **Guess**: Player C output guess $b' \in \{0,1\}$.
5. **Win**: $\text{CPA}_C(\lambda) = \begin{cases} 1 & \text{if } b = b' \\ 0 & \text{if } b \neq b'. \end{cases}$

If $\text{CPA}_C(\lambda) = 1$ then the adversary C guessed correctly which of their two chosen messages was encrypted, based only on observing the ciphertext. Notice that the game requires the player to choose messages of equal length in the 'choose' phase since the ciphertext length always leaks information about the length of the message, namely an upper bound on the message length.

Definition 25 (Semantic security (CPA))**.** An encryption scheme is semantically secure if, for all polynomial-size Boolean circuit families C,

$$|\Pr[\text{CPA}_C(\lambda)] - \frac{1}{2}| = \text{negl}(\lambda).$$

Semantic security means that there exists no algorithm in P/poly that can do more than negligibly better than guessing randomly in determining the message. Semantic security is equivalent to indistinguishability of encryptions (see [Gol04] for proof).

Definition 26 (Indistinguishability of encryptions)**.** An encryption scheme has indistinguishable encryptions if for any key $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$ and any two distinct messages m_1, m_2 of equal length, the ensembles $\{\text{Enc}(pk, m_1)\}_{\lambda\in\mathbb{N}}$ and $\{\text{Enc}(pk, m_2)\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.

Usually, encryption schemes are required to be secure against a stronger type of attack, called chosen-ciphertext attack (CCA). There are two types of CCA attacks; adaptive (called CCA2) and non-adaptive (called CCA1). The CCA1 game is defined exactly like the CPA game but where the player also has oracle access to the decryption algorithm in the choose phase. In other words, the player can decrypt any ciphertexts of their choice before submitting the two messages m_0 and m_1 to the challenger. The CCA2 game is the same as CCA1 except that the player also has oracle access to the decryption algorithm in the guess phase for every ciphertext except the challenge ciphertext. Security against CCA1 and CCA2 attacks are defined analogously to semantic security. Clearly, CCA2 security implies CCA1 security and CCA1 security implies semantic security.

As a consequence of its design, homomorphic encryption schemes cannot be CCA2 secure. The reason is that the player can run the evaluate algorithm on the challenge ciphertext with any circuit of choice and then decrypt the evaluated ciphertext. More formally, consider any challenge ciphertext $\text{Enc}(pk, m_b)$ and the permissable circuit C. Player runs $\text{Eval}(pk, C, \text{Enc}(pk, m_b))$, generating a valid evaluated ciphertext of $C(m_b)$. Player then queries decryption and yields $C(m_b)$. Since C is known to the attacker, information about m_b is leaked. Homomorphic encryption schemes allow the attacker to transform the ciphertext of a message m to a ciphertext of a message related to m by a known function. This property is called *malleability.*

One last security definition that will be relevant later is circular security

Definition 27 (Circular security)**.** A semantically secure homomorphic encryption scheme is circular secure if it is semantically secure even when the adversary is given encryptions of the secret key.

Remark. Circular security is not implied by semantic security because an adversary with a random access oracle cannot efficiently query encryptions of the secret key [Bra18].

5. Hard problems

In this section, we assume every vector is a column vector and we denote a matrix $\mathbf{A}$ with boldface. By $\mathbf{A} \in \mathbb{F}^{n \times m}$ we mean $\mathbf{A}$ is a $n \times m$ matrix with entries in $\mathbb{F}$. Consider a basis $\mathbf{B} = \{b_1, \ldots, b_k\}$. A lattice with basis $\mathbf{B}$ is defined $L(\mathbf{B}) \overset{\text{def}}{=} \{\sum_{i=1}^{k} a_i b_i \mid a_i \in \mathbb{Z}\}$. For any integer $q \geq 2$, we define $\mathbb{Z}_q \overset{\text{def}}{=} (-\frac{q}{2}, \frac{q}{2}] \cap \mathbb{Z}$. For any tuple of integers x (e.g., integer or matrix), we define $[x]_q$ as the tuple of integers over $\mathbb{Z}_q$ such that each element is congruent mod q to the corresponding element in x. For example, $q = 3, x = (5, 1, -2, 6), [x]_q = (-1, 1, 1, 0)$.

5.1 LWE

In 2005, Oded Regev introduced [Reg05] a natural problem; solve a system of modular noisy linear equations. The problem is called learning with errors (LWE) and in 2012, a ring based version called ring-LWE (RLWE) was introduced [LPR12]. Despite being easy to state, the LWE problem turns out to be hard even on average instances. For certain parameter choices, the LWE problem is reducible to extensively studied hard lattice based problems (e.g., GapSVP, SIVP) whereas RLWE is reducible to hard problems on ideal lattices (e.g., ideal-SVP, NTRU). Hardness of these problems are outside the scope of this thesis; we refer the reader to [Reg05; Pei08; Bra+13; LPR12] for more details on hardness results and [Pei15] for a good rundown on hard lattice based problems. Today, essentially all homomorphic encryption schemes are based on LWE and RLWE.

The parameters for LWE are the integers $n = n(\lambda)$ for the dimension, $q = q(n)$ for the global modulus, m for number of samples and a discrete error distribution measure $\chi = \chi(\lambda)$ over $\mathbb{Z}$.

Consider the space of $n \times m$ matrices $\mathbb{Z}_q^{n \times m}$ with uniform distribution, space of secrets $\mathbb{Z}_q^n$ with uniform distribution[1], space of errors $\mathbb{Z}^m$ with discrete distribution χ^m and the

[1]The space of secrets can have distribution χ^n without loss of hardness. See [App+09].

random variable $A_{n,q,\chi,m}$ defined on the direct product as follows

$$\begin{aligned} A_{n,q,\chi,m} \colon \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n \times \mathbb{Z}^m &\to \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m \cong \mathbb{Z}_q^{(n+1)\times m} \\ (\mathbf{A}, s, e) &\mapsto (\mathbf{A}, b^T \overset{\text{def}}{=} [s^T\mathbf{A} + e^T]_q) = \begin{bmatrix} \mathbf{A} \\ b^T \end{bmatrix} \end{aligned}$$

Definition 28 (LWE distribution)**.** The learning with errors distribution is defined as the distribution of $A_{n,q,\chi,m}$

$$\mathrm{LWE}_{n,q,\chi,m} \overset{\text{def}}{=} \mathcal{L}(A_{n,q,\chi,m})$$

In words, sample a $n \times m$ matrix $\mathbf{A}$ with entries in $\mathbb{Z}_q$ at random, choose a uniformly random secret $s \leftarrow \mathbb{Z}_q^n$ and let e consist of m independent errors from the discrete distribution χ. Calculate b^T (i.e, $[s^T\mathbf{A} + e^T]_q$) and output the pair $(\mathbf{A}, b^T)$. The LWE distribution specifies the probability of sampling each pair.

There are two versions of the LWE problem; search-LWE and decision-LWE. Informally, take a sample $x \leftarrow \mathrm{LWE}_{n,q,\chi,m}$ and consider the last row. The decision version is to decide whether the last row is a linear combination of the previous n rows or if it is uniformly random and the search version is to find the explicit linear combination assuming it exists.

Definition 29 (decision-LWE problem)**.** Let $\mathcal{U}_n$ be the uniform distribution on $\mathbb{Z}_q^{(n+1)\times m}$. Construct a PPT algorithm A such that

$$|\Pr[A(\mathcal{U}_n) = 1] - \Pr[A(\mathrm{LWE_{n,q,\chi,m}}) = 1]| > \mathrm{negl}(n)$$

The decision-LWE hardness assumption is that $\mathcal{U}_n$ and $\mathrm{LWE}_{n,q,\chi,m}$ are computationally indistinguishable (i.e., $\mathrm{LWE}_{n,q,\chi,m}$ is pseudorandom).

Definition 30 (search-LWE problem)**.** Let $x = (\mathbf{A}, b^T) \leftarrow \mathrm{LWE}_{n,q,\chi,m}$. Construct a PPT algorithm A such that

$$\Pr[A(x) = s] > \mathrm{negl}(n)$$

The search-LWE hardness assumption is that $\Pr[A(x) = s] = \mathrm{negl}(n)$. There exists a reduction from search-LWE to decision-LWE for $q = \mathrm{poly}(n)$, meaning they are equivalently hard [Bra+13]. We write LWE assumption to refer to hardness of both the search and decision versions.

For the sake of concreteness, let the parameters below be $q = n^2$, $\chi = \mathrm{D}_{\mathbb{Z},\sqrt{n},\vec{0}}(\vec{x})$ be β-bounded discrete Gaussian distribution and $m = 2n \log q$. The LWE problem can be thought of as the matrix $\mathbf{A}$ acting on the secret s as a linear transformation, generating the vector $s\mathbf{A}$ in the rowspace of A as the 'true' solution. The problem is then to find sA given a b in the m dimensional ball with radius specified by error bound, centered at sA.[2] For the below scheme, let $\beta = (\frac{q}{4} - 1)m^{-1}$

Regev's LWE based cryptosystem

In the same paper that the LWE problem was introduced, Regev also described how to construct a simple cryptosystem based on the LWE-assumption. In Regev's cryptosystem, the message is encrypted bit by bit and each bit encrypts to a ciphertext vector in $\mathbb{Z}_q^n$

1. **Key generation**: Generate a uniformly random LWE secret $s' = (s_1, \ldots, s_n) \leftarrow \mathbb{Z}_q^n$ and let $s = (s_1, \ldots, s_n, -1) \in \mathbb{Z}_q^{n+1}$. Using s', generate the $(n+1) \times m$ matrix $\mathbf{A}' \overset{\text{def}}{=} (\mathbf{A}, b^T) \leftarrow \mathrm{LWE}_{n,q,\chi,m}$. Let $\mathrm{KeyGen}(1^\lambda)$ return $\mathbf{A}' \in \mathbb{Z}_q^{(n+1)\times m}$ as the public key and $s \in \mathbb{Z}_q^{n+1}$ as the secret key.
2. **Encryption**: Generate a random 'subset' vector $r \leftarrow \{0,1\}^m$. To encrypt a bit b, compute $c \leftarrow \mathrm{Enc}(\mathbf{A}', b) \overset{\text{def}}{=} [b \cdot \lfloor q/2 \rfloor \cdot (0, \ldots, 0, -1)^T + \mathbf{A}'r]_q \in \mathbb{Z}_q^{n+1}$.
3. **Decryption**: To decrypt, compute $z \overset{\text{def}}{=} [\langle s, c \rangle]_q$ and let $\mathrm{Dec}(s, c)$ be 0 if $|z| < q/4$ and 1 otherwise.

The idea behind the scheme is to embed an encryption of a bit b in the last coordinate of the ciphertext c. To encrypt the bit, first consider a random subset of the m samples and then calculate their sum. Second, if and only if the bit is 1, add $\lfloor q/2 \rfloor$ to the last coordinate (i.e., last row). To see why decryption works, consider an encryption of 0. Then, $\mathrm{Dec}(s, \mathrm{Enc}(\mathbf{A}', 0)) = [\big\langle s, \mathbf{A}'r \big\rangle]_q = [\big\langle s\mathbf{A}', r \big\rangle]_q = [\langle e, r \rangle]_q$. If the bit is 1, then $\mathrm{Dec}(s, \mathrm{Enc}(\mathbf{A}', 1)) = [\big\langle s, (0, \ldots, 0, -\lfloor q/2 \rfloor)^T \big\rangle + \big\langle s, \mathbf{A}'r \big\rangle]_q = [\lfloor q/2 \rfloor + \langle e, r \rangle]_q$. To show correctness, we need to show that $|[\langle e, r \rangle]_q| < q/4$ and $|[\lfloor q/2 \rfloor + \langle e, r \rangle]_q| \geq q/4$. The first inequality holds because $|\,\langle e, r \rangle\,| < \beta m = \frac{q}{4} - 1 < \frac{q}{4}$ and the second inequality holds because $\lfloor q/2 \rfloor + \langle e, r \rangle \in (\lfloor q/2 \rfloor - \frac{q}{4} + 1, \lfloor q/2 \rfloor + \frac{q}{4} - 1) \subset (\frac{q}{4}, \frac{3q}{4}) \overset{[\cdot]_q}{\mapsto} ([\frac{q}{4}, \frac{q}{2}] \cup (-\frac{q}{2}, -\frac{q}{4})) \cap \mathbb{Z}$, all of which are greater than or equal to $q/4$.

The security of the scheme is based on the hardness of two problems; subset-sum and

[2]The knowledgable reader may see the resemblence to the bounded-distance decoding (BDD) problem

LWE. An adversary that obtains r can easily decrypt a ciphertext; simply compute $\mathbf{A}'r$ and compare its last element with the last element of the ciphertext. If they are equal, the bit is 0 and if they are not, the bit is 1. Computing r based on $\mathbf{A}'r$ is essentially the NP-hard problem called the subset-sum problem.

Claim 1. Regev's scheme is semantically secure under the LWE assumption.

Proof. Assume towards contradiction that the scheme is not semantically secure. Then there exists an algorithm C that can distinguish the ensembles $E_0 = \{\text{Enc}(\mathbf{A}', 0)\}_{\lambda \in \mathbb{N}}$ and $E_1 = \{\text{Enc}(\mathbf{A}', 1)\}_{\lambda \in \mathbb{N}}$ with non-negligible advantage ϵ. In other words, $\text{negl}(\lambda) < \epsilon(\lambda) = |\Pr[C(E_0) = 1] - \Pr[C(E_1) = 1]|$. Let $\mathcal{U} = \{\mathcal{U}_n\}_{n(\lambda) \in \mathbb{N}}$ be the uniform distribution ensemble on the corresponding ciphertext space. By triangle inequality, $\epsilon < x + y$ where $x \overset{\text{def}}{=} |\Pr[C(E_0) = 1] - \Pr[C(\mathcal{U}) = 1]|$ and $y \overset{\text{def}}{=} |\Pr[C(\mathcal{U}) = 1] - \Pr[C(E_1) = 1]|$.

Case 1: $x \geq \epsilon/2$. Consider adversary C' that queries and mimics C; $C'(X) = 0$ if $C(X) = 0$ and $C'(X) = 1$ if $C(X) = 1$. Since an LWE sample contains m encryption of 0 under r containing exactly one 1, $\Pr[C'(LWE_{n,q,\chi,m}) = 1] = \Pr[C(E_0) = 1]$ and thus $|\Pr[C'(LWE_{n,q,\chi,m}) = 1] - \Pr[C'(\mathcal{U}) = 1]| = x \geq \epsilon/2 > \text{negl}(\lambda)$

Case 2: $y \geq \epsilon/2$. Consider adversary C'' that translates the input distribution by adding $\lfloor q/2 \rfloor \cdot (0, \dots, 0, -1)^T$ to the last row of each sample and then queries and mimics C like above. For the $\text{LWE}_{n,q,\chi,m}$ distribution, this translation transforms the m encryptions of 0 to encryptions of 1. For the uniform distribution, translation does nothing. Therefore, $\Pr[C''(LWE_{n,q,\chi,m}) = 1] = \Pr[C(E_1) = 1]$ and thus $|\Pr[C''(LWE_{n,q,\chi,m}) = 1] - \Pr[C''(\mathcal{U}) = 1]| = y \geq \epsilon/2 > \text{negl}(\lambda)$

In either case, we have shown there exists an adversary that can distinguish LWE samples from uniform with non-negligible advantage, contradicting the hardness of the LWE assumption. $\square$

In words, if the scheme is not semantically secure then there exists an algorithm C that is either good at distinguishing encryptions of 0 from uniform or encryptions of 1 from uniform. In the first case, the algorithm that mimics C is good at distinguishing LWE samples from uniform. In the second case, the algorithm that translates samples before querying C is good at distinguishing LWE samples from uniform. This contradicts the hardness of the LWE assumption.

Although Regev's scheme has nice natural homomorphic addition by component wise vector addition of ciphertexts, see [Pei15, pp. 36] for details, it is too inefficient for practical purposes. The public key consists of a matrix over Z_q, yielding $O(nm \log q) = O(n^2 \log^2 q) = \tilde{O}(n^2)$ bits and the secret key is $O(n \log q) = \tilde{O}(n)$ bits. The encryption of a message bit is $O(n \log q) = \tilde{O}(n)$ bits, meaning encryption expands bitstrings by a factor of $\tilde{O}(n)$. RLWE address these issues.

5.2 RLWE

The LWE problem is conceptually easy to understand but suffers from expensive overhead. Each element in b is a perturbed linear combination of the secret s and the corresponding column in $\mathbf{A}$, resulting in $O(n)$ operations. To get $m = n$ samples, the total number of operations is $O(n^2)$. In 2012, a more compact ring based learning with errors problem (RLWE) was introduced by Lyubashevsky, Peikert and Regev in [LPR12].

The parameters for RLWE are the integers $n = n(\lambda)$ for the dimension, $q = q(n)$ for the global modulus and a discrete error distribution measure $\chi = \chi(\lambda)$ over $\mathbb{Z}$.

Let the dimension $n = 2^k$ for some natural number k and consider the irreducible polynomial $f(x) = x^n + 1$. Define the polynomial ring $R_q \overset{\text{def}}{=} \mathbb{Z}_q[x]/\langle f(x) \rangle$. The ring R_q is viewed as the ring of polynomials with coefficients in $\mathbb{Z}_q$, having degree less than n. There is a natural correspondence between elements of R_q and $\mathbb{Z}_q^n$ where each polynomial coefficient corresponds to an element in the vector. Let the error space R'_q be R_q endowed with error distribution χ^n over the integer coefficients and consider the random variable $A'_{n,q,\chi}$ defined as follows

$$\begin{aligned} A'_{n,q,\chi} \colon R_q \times R_q \times R'_q &\to R_q^2 \\ (a, s, e) &\mapsto (a, b \overset{\text{def}}{=} [a \cdot s + e]_q) \end{aligned}$$

Definition 31 (RLWE distribution). The learning with errors distribution is defined as the distribution of $A'_{n,q,\chi}$

$$\mathrm{RLWE}_{n,q,\chi} \overset{\text{def}}{=} \mathcal{L}(A'_{n,q,\chi})$$

In words, sample a uniformly random polynomial $a \in R_q$, a uniformly random secret $s \in R_q$ and a random error $e \in R_q$ from the discrete distribution χ^n. Calculate b (i.e, $[a \cdot s + e]_q$) and output the pair (a, b). The RLWE distribution specifies the probability of sampling each pair.

The decision-RLWE problem is to distinguish between the RLWE distribution and the uniform distribution on R_q^2 and the search problem is to find the secret s given a sample $(a, b) \leftarrow \text{RLWE}_{n,q,\chi}$.

Definition 32 (decision-RLWE problem)**.** Let $\mathcal{U}_n$ be the uniform distribution on R_q^2. Construct a PPT algorithm A such that

$$|\Pr[A(\mathcal{U}_n) = 1] - \Pr[A(\mathrm{RLWE_{n,q,\chi}}) = 1]| > \mathrm{negl}(n)$$

The decision-RLWE hardness assumption is that $\mathcal{U}_n$ and $\text{RLWE}_{n,q,\chi}$ are computationally indistinguishable (i.e., $\text{RLWE}_{n,q,\chi}$ is pseudorandom).

Definition 33 (search-RLWE problem)**.** Let $x = (a, b) \leftarrow \text{RLWE}_{n,q,\chi}$. Construct a PPT algorithm A such that

$$\Pr[A(x) = s] > \mathrm{negl}(n)$$

The search-RLWE hardness assumption is that $\Pr[A(x) = s] = \mathrm{negl}(n)$.

Note that in LWE, the sample size m is embedded as a parameter while in RLWE, adversary generates $m = \mathrm{poly}(\lambda)$ samples from the $\text{RLWE}_{n,q,\chi}$ themselves.

In LWE, the b vector is of size m. Each element is calculated by an inner product of the secret s and the corresponding column in $\mathbf{A}$, resulting in $O(n)$ operations. To get $m = n$ samples, the total number of operations is $O(n^2)$. For RLWE, the b vector is a polynomial with n coefficients. By using the fast fourier transform (FFT) algorithm, polynomial multiplication can be calculated in $O(n \log n)$ operations. To get n samples, only one polynomial multiplication is needed, resulting in $O(n \log n)$ operations. As a consequence, the cost of generating public keys is reduced from $O(n^2)$ to $O(n \log n)$ by exploiting the structure of the ring.

6. History of fully homomorphic encryption

The history of homomorphic encryption began with Rivest, Adleman and Dertouzos [RAD78] already in 1978 where they identified the use of delegating computation to a third party. They were not successful in constructing a secure scheme that supports both multiplication and addition of ciphertext, as required for arbitrary function evaluation. The problem of finding a secure scheme that supports both operations turned out to be difficult and remained unsolved for more than 30 years. Interestingly, multiple secure scheme that supports either multiplication or addition of ciphertexts were discovered. For example, the RSA scheme supports multiplication of ciphertexts and the Paillier scheme supports addition of ciphertexts. It was not until 2009, when Craig Gentry published his PhD thesis [Gen09], that a fully homomorphic encryption scheme was first constructed. Gentry's original scheme was inefficient and since then, many improved schemes have been introduced, both in terms of efficiency and security. In this chapter, we will go through the history of fully homomorphic encryption, split into four generations.

6.1 First generation

Gentry scheme is based on ideal lattices. The construction considers a polynomial ring over the integers modulo an ideal generated by a cyclotomic polynomial, similar to the RLWE setup. Then, there exists an ideal with norm polynomially bounded with respect to the dimension such that the ideal generates a lattice (see [Gen09] for details). The security of Gentry's scheme was based on three hardness assumptions: sparse subset-sum problem (SSSP), bounded-distance decoding problem (BDD) and the ideal shortest vector problem (ideal-SVP). The original decryption function in Gentry's scheme was not bootstrappable (see definition 34), without the use of a method he introduced, called squashing. The idea was to include extra information in the public key which allowed for easier decryption, which meant that security also required the SSSP hardness assumption.

The Gentry's scheme was improved by Smart and Vercauteren in [SV09] where they introduced batching of multiple plaintexts encrypted into a single ciphertext using the Chinese Remainder Theorem (ciphertext packing is the same idea where multiple ciphertexts are packed together). Gentry and Halevi implemented the scheme in [GH10], including the use of the batching technique and an optimized squashing technique to bring down the degree of the decryption polynomial from hundreds (estimated by Smart and Vercauteren) to 15. In Gentry's original scheme, the key generation algorithm was impractically slow. In their implementation, they reduced the asymptotic complexity to $\tilde{O}\left(n^{1.5}\right)$ for cyclotomic fields where the order of the root of unity is a power of 2 (i.e., $\mathbb{Z}[x]/\langle x^n + 1\rangle$ for $n = 2^k$). Scholl and Smart extended the implementation to arbitrary cyclotomic fields in [SS11].

In 2010, Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan finalized a new scheme (DGHV scheme) that was based on multiplication and addition over the integers as opposed to over polynomial rings [Dij+09]. The hardness of the DGHV scheme was based on the approximate greatest common divisor problem (AGCD), and the SSSP due to squashing of the decryption circuit. Subsequent works optimized the DGHV scheme through modulus switching in [CNT11] and plaintext batching in [Kim+13] and [CLT13].

6.2 Second generation

The second generation began in 2011 with Brakerski and Vaikuntanathan [BV11]. Their paper introduced two main contributions. Their first contribution was to base the hardness of the proposed BV scheme on the well known LWE problem, which means using arbitrary lattices instead of ideal lattices. Their second contribution was the removal of squashing from previous schemes (Gentry and Halevi, independently, also managed to remove squashing in [GH11]). This meant that the BV scheme did not require the SSSP hardness assumption. As an optimization technique, the paper was the first to introduce what would later be called *modulus switching*. Modulus switching is an alternative to Bootstrapping in managing noise growth (see Chapter 7) by scaling down the ciphertext noise and the global modulus, effectively switching the modulus of the scheme. Modulus switching keeps the ratio of noise to modulus the same but is still effective since the magnitude of the noise, and thus also its growth rate, is smaller. A downside of the BV scheme (and the rest of the schemes based on LWE in the second generation) was that an expensive step called re-linearization is necessary to prevent homomorphic multiplication causing ciphertext length from growing exponentially. In the BV scheme, homomorphic multiplication corresponds to a tensor product, changing the structure of the ciphertext. In order to re-linearize the result, the scheme uses key-switching. This requires applying a large re-linearization matrix of size $\Omega(n^3)$ (embedded in the public key) to the tensorproduct.

In a follow up paper in 2011, Brakerski, Gentry and Vaikuntanathan introduced the BGV scheme that showed, for the first time, bootstrapping was not necessary for a fully homomorphic encryption scheme. Their construction worked for both LWE and RLWE, and ironically used bootstrapping as an optimization technique. Brakerski later modified the LWE based BGV scheme by replacing the modulus switching technique with a similar *scale invariant* approach [Bra12]. The idea was to scale down the ciphertext by the constant global modulus q, thus reducing the noise bound $\beta < q$ to a corresponding fraction, mod 1. Brakerski showed that noise growth for scale invariant ciphertexts under homomorphic multiplication only grows polynomially w.r.t dimension n (and hence security parameter). Fan and Vercauteren converted Brakerski's scale invariant scheme to the RLWE setting, resulting in the BFV scheme [FV12]. It is worth mentioning that as a part of the second generation of homomorphic encryption, there were schemes based on the NTRU public key encryption scheme from 1998 [HPS98]. However, these schemes required parameter sizes larger than originally proposed due to vulnerabilities from subfield lattice attacks [ABD16].

6.3 Third generation

In 2013, Gentry, Sahai and Waters introduced a new scheme called the GSW scheme [GSW13]. The GSW scheme was also based on LWE, but unlike its predecessors, it did not require the re-linearization step. This is because the scheme operates on matrices which has an inherent natural addition and multiplication operation. Since the relinearization matrix is no longer needed, the space complexity of GSW is quasi-quadratic as opposed to quasi-cubic for BGV and BFV. Furthermore, GSW also did not require bootstrapping to achieve FHE. A downside of the GSW scheme is the complexity of ciphertext matrix multiplication, yielding in slower multiplication than the ring based second generation schemes.

A defining trait of the third generation is the efficiency gain from bootstrapping. There were two main new faster bootstrapping algorithms. Jacob Alperin-Sheriff and Chris

Peikert [AP13] introduced bootstrapping of non-packed ciphertexts in quasilinear time w.r.t the security parameter (which is important in the third generation as there is no packing) and Gama et al. [Gam+14] built on their work and introduced a new type of homomorphic gate. Léo Ducas and Daniele Micciancio developed a scheme called FHEW that used Alperin-Sheriff and Peikerts bootstrapping algorithm to allow for much faster bootstrapping using what is now called programmable bootstrapping [DM14]. Another paper introuced a scheme based on the LWE over the torus, meaning a stricter security assumption, called TFHE [Chi+18] which allowed for bootstrapping in 0.1 milliseconds after subsequent optimizations [MP20].

6.4 Fourth generation

The fourth generation of homomorphic encryption schemes began in 2016 when Jung Hee Cheon, Andrey Kim, Miran Kim and Yongsoo Song introduced a new RLWE based leveled FHE scheme called CKKS [Che+16] which they subsequently turned into a pure FHE scheme through bootstrapping in [Che+18]. CKKS is based on approximate arithmetic, allowing for computation on real and complex numbers. To achieve this, a vector of numbers (real or complex) are encoded using rounding as a integral plaintext polynomial in a cyclotomic polynomial ring. This is a fundamentally different approach as the scheme operates on an approximation of the message as opposed to the real message. The CKKS scheme is useful for applications concerning floating point numbers, such as machine learning and neural networks. Subsequent works focused on optimizing the bootstrapping and ciphertext packing techniques.

In 2020, Baiyu Li and Daniele Micciancio showed that the CKKS scheme was vulnerable to passive attacks with respect to IND-CPA security [LM20]. To mitigate this issue, a new security notion called IND-CPA+ was introduced. IND-CPA+ and IND-CCA1 differ in that the adversary is only allowed to query decryption on evaluated ciphertexts, as opposed to arbitrary ciphertexts. Li and Micciancio showed that IND-CPA+ is eequivalent to IND-CPA for exact encryption schemes but strictly stronger for approximate schemes like CKKS.

7. Noise management

The main problem with homomorphic encryption schemes is the noise. Noise is introduced in the ciphertexts during the encryption process and when the ciphertexts undergo homomorphic operations, the noise grows. After sufficiently many operations, the noise grows to the point where the decryption of the evaluated ciphertext fails. In order to reach FHE, it is necessary to control the noise to allow for sufficient number of operations. Noise management is the process of controlling the noise during homomorphic evaluation. In his 2009 seminal PHD thesis [Gen09], Craig Gentry showed that FHE was possible by using a noise management technique called Bootstrapping. Bootstrapping is an algorithm that transforms a possibly noisy ciphertext into a correct evaluated ciphertext with little noise by using an encryption of the secret key to decrypt the noisy ciphertext homomorphically.

Throughout this chapter, ciphertext are hard-wired into decryption algorithms, meaning that each decryption algorithm is nessesarily correct for only one specified ciphertext. This is to simplifying notation only. It is equivalent to require only one decryption algorithm, passing any ciphertext as input. The difference is that the message need to be encrypted twice (possibly under different keys) in the latter case, since the final decryption removes one layer of encryption on both arguments resulting in a once encreypted message and pure secret key as required. Some authors, including Gentry in his original paper, specify bootstrapping in this way.

7.1 Key-Switching

This section is based on [Bra18].

As a natural segway into bootstrapping, we first introduce a related but different concept called key-switching. Since HE schemes are designed with the constraint of supporting homomorphic operations, they are often inefficient. Therefore, it would be desirable to use a more efficient non-homomorphic scheme to encrypt the large messages, but still retaining the homomorphic property. Key-switching is a technique that allows for homomorphic computation on ciphertexts encrypted under a non-homomorphic scheme.

In particular, it allows for transforming a ciphertext under a non-HE scheme to a corresponding ciphertext under a HE scheme. The idea is to encrypt the much shorter secret key of the non-HE scheme under the public key of the HE scheme and hard-wire the ciphertexts into the function of interest. Let (H.KeyGen, H.Enc, H.Dec, Eval) be a homomorphic encryption scheme and (KeyGen, Enc, Dec) be a non-homomorphic encryption scheme. Let $(hpk, hsk) \leftarrow \mathrm{H.KeyGen}(\lambda)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}(\lambda)$. Consider computable function C, the vector of ciphertexts $c \leftarrow \mathrm{Enc}(pk, m)$ and an encrypted secret key $sk' \leftarrow \mathrm{H.Enc}(hpk, sk)$. Define $\hat{C}_c(\cdot) \overset{\text{def}}{=} C(\mathrm{Dec}(\cdot, c))$. $C(m)$ can be computed homomorphically by decrypting $\mathrm{Eval}(hpk, \hat{C}_c, sk')$. Indeed, this is a correct encryption of $C(m)$ since

$$\mathrm{Dec}(hsk, \mathrm{Eval}(hpk, \hat{C}_c, sk')) = \hat{C}_c(sk) = C(\mathrm{Dec}(sk, c)) = C(m)$$

We have shown that it is possible to use a non-homomorphic encryption scheme to encrypt the message and still perform homomorphic computation on it. Key-switching transforms a ciphertext encrypted under a non-HE scheme to an evaluated ciphertext encrypted under a HE scheme. The underlying assumption is that the HE scheme can evaluate $\hat{C}_c$, meaning it can first run the decryption algorithm and then compute the desired function C to generate a valid evaluated ciphertext. Even under the assumption that the decryption algorithm is simple enough to be evaluated by the scheme, a general C consists of several multiplication and addition gates, meaning it is unlikely that the scheme can evaluate $\hat{C}_c$. However, if it is possible to split C into smaller components, $C = C^m \circ \cdots \circ C^1$, and evaluate each component separately, $c_{(i)} = \mathrm{Eval}(hpk, \hat{C}^i_{c_{(i-1)}}, sk')$, where $c_{(0)} \leftarrow Enc(pk, m)$, computing C homomorphically is achievable assuming $\hat{C}^i_{c_{(i-1)}}$ is permissible for all i. As the construction currently stands, further computation on the ciphertext is not possible. To see why, consider $c_1 = \mathrm{Eval}(hpk, \hat{C}^1_{c_{(0)}}, sk')$ and let $c_2 = \mathrm{Eval}(hpk, \hat{C}^2_{c_1}, sk')$. We hope decrypting c_2 yields $(C^2 \circ C^1)(m)$, but $\mathrm{Dec}(hsk, c_2) = \mathrm{Dec}(hsk, \mathrm{Eval}(hpk, \hat{C}^2_{c_1}, sk')) = C^2(\mathrm{Dec}(sk, c_1))$. However, since c_1 is encrypted under the HE scheme, the non-homomorphic decryption algorithm applied to c_1 does not make sense and decryption fails.

7.2 Bootstrapping

Key-switching is a nice optimization technique for encrypting messages faster, but it does not allow for further computation on the ciphertext. The requirement is that the

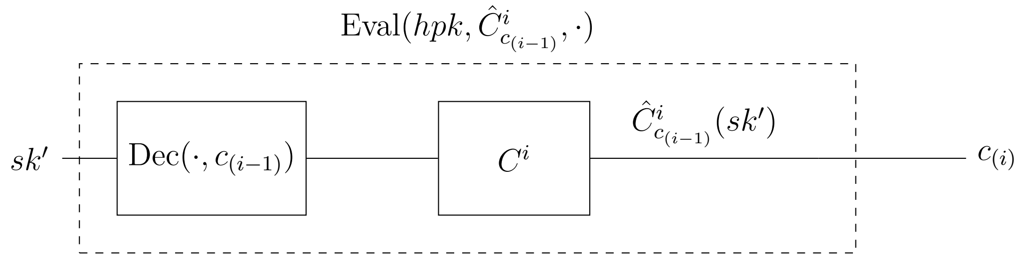
$\text{Eval}(hpk, \hat{C}^i_{c_{(i-1)}}, \cdot)$
sk'
$\text{Dec}(\cdot, c_{(i-1)})$
C^i
$\hat{C}^i_{c_{(i-1)}}(sk')$
$c_{(i)}$

Figure 3: Partial homomorphic computation of $C = C^m \circ \cdots \circ C^i \circ \cdots \circ C^1$ using bootstrapping. For $i = m$ the output decrypts to $C(m)$.

decryption algorithm and the secret key originate from the HE scheme. We therefore only consider the HE scheme. Let (KeyGen, Enc, Dec, Eval) be a HE scheme, $(pk, sk) \leftarrow \text{KeyGen}(\lambda)$ and $sk' \leftarrow \text{Enc}(pk, sk)$. Since this construction encrypts the secret key using its own public key, circular security is assumed (for now). Under this construction, decryption of c_1 is still correct as $\text{Dec}(sk, c_1) = C^1_{c_{(0)}}(sk) = C^1(\text{Dec}(sk, c_{(0)})) = C^1(m)$, but the difference is that decryption of $c_{(i)}$ also works since, by induction,

$$\text{Dec}(hsk, \text{Eval}(hpk, \hat{C}^i_{c_{(i-1)}}, sk')) = \hat{C}^i_{c_{(i-1)}}(sk) = C^i(\text{Dec}(sk, c_{(i-1)})) = C^i \circ C^{i-1} \circ \cdots \circ C^1(m)$$

In essence, we have constructed an algorithm to evaluate C on a ciphertext c as follows: The first step is to split C into m smaller components, $C = C^m \circ \cdots \circ C^1$. The second step is to run the encrypted secret key through the decryption circuit for the ciphertext. The third step is to evaluate the first component on the ciphertext output of the previous step and let this be the ciphertext. The fourth step is to repeat the second and third step for a total of m times, incrementing the component each time. The final ciphertext decrypts to $C(m)$. One iteration of step 2 and 3 is illustrated in Figure 3.

Splitting the desired function C into multiple components $C^1, \ldots, C^m$ is the key insight into constructing FHE. Consider the simplest case; each component function is either a multiplication gate or an addition gate. In other words, for a vector of ciphertexts $c = \langle c_1, c_2 \rangle$, define a multiplicative component circuit of C as $C^i(c) = c_1 \times c_2$, $\hat{C}^i_c(\cdot) = C^i(\text{Dec}(\cdot, c)) = C^i(\langle \text{Dec}(\cdot, c_1), \text{Dec}(\cdot, c_2) \rangle) = \text{Dec}(\cdot, c_1) \times \text{Dec}(\cdot, c_2)$ and where the addition case is analogous. Note that $\hat{C}^i_c(sk) = m_1 \times m_2$. It turns out that if the scheme can evaluate just these two types of circuits, then it can evaluate every circuit. This is the idea behind bootstrapping.

Definition 34 (Bootstrappable encryption scheme)**.** Let $\mathcal{E}$ be a $\mathcal{C}$-homomorphic encryption scheme. Consider the following *augmented decryption circuits* for $\mathcal{E}$:

$$B^{(m)}_{c_1,c_2}(\cdot) \overset{\text{def}}{=} \mathrm{Dec}(\cdot, c_1) \times \mathrm{Dec}(\cdot, c_2)$$
$$B^{(a)}_{c_1,c_2}(\cdot) \overset{\text{def}}{=} \mathrm{Dec}(\cdot, c_1) + \mathrm{Dec}(\cdot, c_2)$$

$\mathcal{E}$ is *bootstrappable* if

$$\{B^{(m)}_{c_1,c_2}(\cdot), B^{(a)}_{c_1,c_2}(\cdot) \mid c_1, c_2 \in \mathcal{Y}\} \subset \mathcal{C}$$

The augmented decryption circuits have the ciphertexts hard-wired into its description and takes as input an encryption of the secret key. A bootstrappable scheme correctly evaluates the set of all augmented decryption circuits. In particular, it correctly evaluates its own decryption circuit by letting c_2 be an encryption of the multiplicative or additive identity respectively.

Theorem 7 (Gentrys bootstrapping theorem, simplified by Vaikuntanathan [Gen09; Vai11])**.** *Any bootstrappable scheme can be transformed into a leveled FHE. Furthermore, if circular security holds, it can be transformed into a pure FHE scheme.*

Remark. Bootstrapping is sufficient for leveled FHE, but not necessary. See [BGV14] for a leveled FHE scheme that does not require bootstrapping.

Proof. Let (KeyGen, Enc, Dec, Eval) be a bootstrappable scheme. Assume first that circular security holds. We construct a pure FHE scheme (KeyGen', Enc', Dec', Eval') as follows:

1. **Key generation**: Generate a key pair $(\hat{pk}, sk)$ using KeyGen(1^λ). Let $sk' \leftarrow$ Enc$(\hat{pk}, sk)$. Define $pk = (\hat{pk}, sk')$ and let KeyGen'(1^λ) return (pk, sk).
2. **Encryption**: Let Enc' be the same as Enc.
3. **Decryption**: Let Dec' be the same as Dec.
4. **Evaluation**: Let Eval'(pk, C, c) transform input circuit C as follows: For each layer of the circuit, consider the gate taking ciphertexts c_i, c_j. If it is a multiplication gate, swap it with $B^{(m)}_{c_i,c_j}(\cdot)$ and if it is an addition gate, swap it with $B^{(a)}_{c_i,c_j}(\cdot)$. Run the encrypted secret key[1] through the augmented decryption circuit and let the outputs act as ciphertext inputs to the next layer. Repeat for all layers of C and denote the transformed circuit C'. Let Eval'(pk, C, c) be the vector of ciphertexts

[1]The encrypted secret key can be seen as an advice wire over the circuit, assessible at any layer

at the output wires of C'.

To see why decryption is correct, remark that the scheme can evaluate each augmented decryption circuit. Therefore, the output ciphertexts decrypts to the gate applied to the decryption of the input ciphertexts. By induction, the input ciphertexts in turn also decrypts correctly since the input layer to the circuit is fresh ciphertexts. More formally, consider circuit output k, denoted $C(m)_k$, undergoing a (say multiplication) operation in the last layer. Then, $C(m)_k = C_1(m) \times C_2(m)$ for some subcircuits C_1, C_2 of C. Let ciphertext c be input to the transformed circuit C' and assume that, by induction, input ciphertexts for last layer z_1, z_2 satisfies $\mathrm{Dec}(sk, z_1) = C_1(m)$ and $\mathrm{Dec}(sk, z_2) = C_2(m)$. Then, decryption of $C'(c)_k$ yields

$$\begin{aligned}\mathrm{Dec}(sk, \text{Eval'}(pk, C, c)_k) &= \mathrm{Dec}(sk, B^{(m)}_{z_1,z_2}(sk')) = B^{(m)}_{z_1,z_2}(sk) \\ &= \mathrm{Dec}(sk, z_1) \times \mathrm{Dec}(sk, z_2) = C_1(m) \times C_2(m) = C(m)_k\end{aligned}$$

Security of the scheme holds by the fact that the original scheme is secure and circular security hold, meaning the encrypted secret key under its own public key does not compromise the security. In other words, the evaluate algorithm is only using public information to evaluate the circuit. Since the circuit was arbitrary, the scheme is a pure FHE scheme. Assume now that circular security does not hold. We construct a leveled FHE scheme (KeyGen', Enc', Dec', Eval') as follows:

1. **Key generation**: Given input parameters $(1^\lambda, 1^L)$, generate $L + 1$ key pairs $(\hat{pk}_i, sk_i)_{i=0,\dots,L}$ using KeyGen. Let $sk'_i \leftarrow \mathrm{Enc}(\hat{pk}_{i+1}, sk_i)$ for all $i = 0, \dots, L-1$. Define $sk = (sk_0, \dots, sk_L)$ and define $pk = (pk_0, sk'_0, pk_1, sk'_1, \dots, sk'_{L-1}, pk_L)$ and let KeyGen' return (pk, sk).

2. **Encryption**: Let Enc' be the same as Enc.

3. **Decryption**: Let Dec' be the same as Dec.

4. **Evaluation**: Let Eval' take input circuit of depth at most L and 'pad' it so that it has depth exactly L. Transform the circuit as follows: For layer $i = 1, \dots, L$ of the circuit, consider the gate taking ciphertexts c_i, c_j. If it is a multiplication gate, swap it with $B^{(m)}_{c_i,c_j}(\cdot)$ and if it is an addition gate, swap it with $B^{(a)}_{c_i,c_j}(\cdot)$. Run the encrypted secret key sk'_{i-1} through the augmented decryption circuits and let the outputs act as ciphertext inputs to the next layer. Let Eval'(pk, C, c) be the vector of ciphertexts at the output wires of C'.

In the transformed circuit C', the input ciphertexts to layer i is encrypted under public key $i-1$. In other words, to decrypt the ciphertexts resulting from layer i, we use private key sk_i. Since the circuit has exactly L layers, the last layer is decrypted using sk_L. Using the same notation as before, the correctness of the decryption algorithm follows by same argument:

$$\begin{aligned}\text{Dec}(sk_L, \text{Eval'}(pk, C, c)_k) &= \text{Dec}(sk_L, B^{(m)}_{z_1,z_2}(sk'_{L-1})) = B^{(m)}_{z_1,z_2}(sk_{L-1}) \\ &= \text{Dec}(sk_{L-1}, z_1) \times \text{Dec}(sk_{L-1}, z_2) \\ &= C_1(m) \times C_2(m) = C(m)_k\end{aligned}$$

The security of the scheme follows by the fact the each encrypted secret key is encrypted under the public key of the next layer, meaning that it is computationally indistinguishable from a random ciphertext. In other words, the secret keys are not encrypted under their own public keys, avoiding circular security assumption. Since the circuit was arbitrary, the scheme is a leveled FHE scheme. $\square$

The difference between the pure FHE scheme and the leveled FHE scheme is that in the former, there is only need for one key pair (pk, sk), where the encrypted secret key can be made public (by circular security assumption) and used to bootstrap in every layer. This implies the transformed circuit can have arbitrary depth since each layer outputs a valid evaluated ciphertext. In the latter, circular security does not hold and the encrypted secret key cannot be reused. Therefore, every layer needs a new encrypted secret key. In general, any scheme generating a fixed amount of valid keys pairs can only bootstrap finitely many times. Since ciphertext noise grows for each operation, the depth of the circuit has to be bounded. In particular, for a set of circuits with depth at most L, $L+1$ valid keypairs is sufficient.

Noise management in practice

As of today, pure FHE requires circular security. Circular security assumption improves the efficiency of the scheme since the key generation algorithm is run only once. However, the construction is still extremely inefficient. Bootstrapping is an expensive operation as it involves constructing and evaluating a decryption circuit. In practice, bootstrapping is only done when necessary, meaning when the noise is about to cause decryption to fail. When this is about to happen, the ciphertext $c \leftarrow \text{Enc}(pk, m)$ is "refreshed" by replacing it with the less noisy $\text{Dec}(sk', c)$. Note that this is essentially an augmented decryption circuit where the other ciphertext is an encryption of the multiplicative (or additive) identity. Indeed, $\mathrm{Dec}(sk, \mathrm{Dec}(\cdot, c), sk') = \mathrm{Dec}(sk, c) = m$.

- Talk about the intuition of bootstrapping - Explain that bootstrapping is essentially only a refreshing of the ciphertext

8. Implementation

In this chapter we will present a fully homomorphic encryption scheme, called GSW, developed by Gentry, Sahai and Waters [GSW13]. The GSW scheme is based on approximate eigenvectors. The appeal of the scheme is its relative conceptual simplicity due to natural matrix addition and matrix multiplication operations. The scheme is leveled FHE even without bootstrapping but, just like every existing FHE scheme, requires bootstrapping and circular security assumption for pure FHE. Security is based on the LWE hardness assumption. Throughout this chapter, all parameters are chosen from LWE as before and the norm is assumed to be $\|A\| = \|A\|_\infty \overset{\text{def}}{=} \max_i \sum_j |a_{ij}|$ for matrix or vector A (manhattan norm for vector). Similar to Regev's scheme, the messages in GSW are bits. The ciphertexts in GSW are matrices where the secret key can be thought of as an approximate left eigenvector with the message bit as eigenvalue. Ideally, for a square ciphertext matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times n}$ and secret vector $s \in \mathbb{Z}_q^n$, a scheme should satisfy $s^T\mathbf{C} = bs^T + e^T$ for some small error vector e and some message bit b. Unfortunately however, multiplication in this scheme leads to rapid error growth:

$$\begin{aligned} s^T(\mathbf{C}_1 \times \mathbf{C}_2) &= (s^T\mathbf{C}_1)\mathbf{C}_2 = b_1 s^T\mathbf{C}_2 + e_1^T\mathbf{C}_2 \\ &= b_1(b_2 s^T + e_2^T) + e_1^T(\mathbf{C}_2) = b_1 b_2 s^T + b_1 e_2^T + e_1^T\mathbf{C}_2 \end{aligned}$$

Notice that the term $e_1^T\mathbf{C}_2$ in the error is bounded by $n\frac{q}{2}\|e_1\|$, for some large q. The problem is that $\|\mathbf{C}_2\|$ has a large bound. If however, $\mathbf{C}_2$ is instead defined over $\{-1, 0, 1\}$, then the new total error satisfies $\|b_1 e_2^T + e_1^T\mathbf{C}_2\| \leq \|b_1 e_2^T\| + \|e_1^T\mathbf{C}_2\| \leq \|e_2^T\| + n\|e_1^T\| \leq (n+1)\max\{\|e_1^T\|, \|e_2^T\|\}$. This suggests transforming ciphertext matrices to lower norm.

Define $l \overset{\text{def}}{=} \lceil \log q \rceil$ and define the matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nl}$ as follows:

$$\mathbf{G} \overset{\text{def}}{=} \begin{bmatrix} 1 & 2 & \cdots & 2^{l-1} & & & & & & & & \\ & & & & 1 & 2 & \cdots & 2^{l-1} & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & 1 & 2 & \cdots & 2^{l-1} \end{bmatrix}$$

Define the function $G^{-1}\colon \mathbb{Z}_q^n \to \{-1,0,1\}^{nl}$ mapping an input column vector to a low norm, high dimension, decomposition column such that $\mathbf{G}G^{-1}(x) = x$ for all $x \in \mathbb{Z}_q^n$. For a matrix input $\mathbf{A} = [a_1|a_2|\dots|a_m]$, we define $G^{-1}(\mathbf{A}) \overset{\text{def}}{=} [G^{-1}(a_1)|G^{-1}(a_2)|\dots|G^{-1}(a_n)]$.

To make the dimensions match, we want the secret key $s \in \mathbb{Z}_q^n$ to be an approximate eigenvector in the sense that for any (fresh or evaluated) ciphertext $\mathbf{C} \in \mathbb{Z}_q^{n\times nl}$ encrypting bit b, the following *invariant equation* holds:

$$s^T\mathbf{C} = b(s^T\mathbf{G}) + e^T$$

for some error $e \in \mathbb{Z}_q^{nl}$. Note that the error in the invariant equation always is of size nl, not to be confused with the error in the LWE samples that is of size m. To allow for homomorphic operations, we need to show how to add and multiply ciphertexts such that the corresponding bits are added and multiplied and that the invariant equation is preserved. In the GSW scheme, we define homomorphic addition as the normal matrix addition operation and homomorphic multiplication as the first ciphertext matrix multiplied with the decomposition of the second ciphertext matrix. The invariant equation is preserved under addition and multiplication:

$$\begin{aligned}
s^T(\mathbf{C}_1 + \mathbf{C}_2) &= s^T\mathbf{C}_1 + s^T\mathbf{C}_2 = b_1 s^T\mathbf{G} + e_1^T + b_2 s^T\mathbf{G} + e_2^T \\
&= (b_1 + b_2)s^T\mathbf{G} + (e_1^T + e_2^T) \\
s^T(\mathbf{C}_1 \times G^{-1}(\mathbf{C}_2)) &= s^T\mathbf{C}_1 \times G^{-1}(\mathbf{C}_2) = (b_1 s^T\mathbf{G} + e_1^T) \times G^{-1}(\mathbf{C}_2) \\
&= b_1 s^T\mathbf{G}G^{-1}(\mathbf{C}_2) + e_1^T G^{-1}(\mathbf{C}_2) = b_1 s^T\mathbf{C}_2 + e_1^T G^{-1}(\mathbf{C}_2) \\
&= b_1(b_2 s^T\mathbf{G} + e_2^T) + e_1^T G^{-1}(\mathbf{C}_2) = b_1 b_2 s^T\mathbf{G} + b_1 e_2^T + e_1^T G^{-1}(\mathbf{C}_2)
\end{aligned}$$

Thus, both addition and multiplication preserve the invariant equation. For addition, the noise growth is bounded by $2\max\{\|e_1^T\|, \|e_2^T\|\}$ whereas for multiplication, $\|b_1 e_2^T + e_1^T G^{-1}(\mathbf{C}_2)\| \leq \|e_2^T\| + \|e_1^T G^{-1}(\mathbf{C}_2)\| \leq \|e_2^T\| + nl\|e_1^T\| \leq (nl+1)\max\{\|e_1^T\|, \|e_2^T\|\}$. Notice that noise growth from multiplication is larger than that of addition in the worst case.

8.1 GSW scheme

In this section, we present a leveled FHE version of the GSW scheme based on the work [Hal17]. The message is a bit b and the ciphertext is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times nl}$. The security is based on the LWE hardness assumption and shows high similarity to Regev's scheme. The error distribution is the discrete Gaussian $\chi = \mathrm{D}_{\mathbb{Z},\sqrt{n},\vec{0}}$ with the error bound $\beta = n$. Notice that all parameters n, q, χ, m are functions polynomially bounded by both the security parameter λ and depth parameter L.

1. **Key generation**: Generate a uniformly random LWE secret $s' = (s_1, \ldots, s_{n-1}) \leftarrow \mathbb{Z}_q^{n-1}$ and let $s = (s_1, \ldots, s_{n-1}, s_n \overset{\text{def}}{=} -1) \in \mathbb{Z}_q^n$. Using s', generate the $n \times m$ matrix $\mathbf{A}' \overset{\text{def}}{=} (\mathbf{A}, b^T) \leftarrow \mathrm{LWE}_{n-1,q,\chi,m}$. If $\|e^T\| = \|[s^T\mathbf{A}']_q\|$ is not bounded by n, then resample (note that resampling is "rare" as $\mathbb{E}[\|e^T\|]) \approx \sqrt{n}$). Let KeyGen$(1^\lambda, 1^L)$ return $\mathbf{A}' \in \mathbb{Z}_q^{n\times m}$ as the public key and $s \in \mathbb{Z}_q^n$ as the secret key.
2. **Encryption**: Generate a random 'subset' matrix $\mathbf{R} \leftarrow \{0,1\}^{m\times nl}$. To encrypt a bit b, compute $\mathbf{C} \leftarrow \mathrm{Enc}(\mathbf{A}', b) \overset{\text{def}}{=} [b \cdot \mathbf{G} + \mathbf{A}'\mathbf{R}]_q \in \mathbb{Z}_q^{n\times nl}$.
3. **Evaluation**: For a given arithmetic circuit C, swap each addition gate to the standard matrix addition function $(\mathbf{C}_1, \mathbf{C}_2) \mapsto \mathbf{C}_1 + \mathbf{C}_2$ and swap each multiplication gate to $(\mathbf{C}_1, \mathbf{C}_2) \mapsto \mathbf{C}_1 \times G^{-1}(\mathbf{C}_2)$
4. **Decryption**: Define $w \overset{\text{def}}{=} (0, 0, \ldots, 0, -\lfloor q/2 \rfloor) \in \mathbb{Z}_q^n$. To decrypt, compute $z \overset{\text{def}}{=} [s^T \times \mathbf{C} \times G^{-1}(w)]_q$ and let Dec(s, c) be 0 if $|z| < q/4$ and 1 otherwise.

The encrypted bit satisfies the invariant equation since $s^T\mathbf{C} = bs^T\mathbf{G} + s^T\mathbf{A}' \times \mathbf{R} = bs^T\mathbf{G} + e^T\mathbf{R}$ where the ciphertext error $e^T\mathbf{R}$ is bounded by $m\|e^T\| \leq m \times n$ from KeyGen process. Furthermore, the evaluated ciphertext also satisfies the invariant equation and the noise grows by at most by a factor $nl + 1$ per layer. Therefore, for any circuit C, the error at depth d is bounded by $nm(nl + 1)^d$.

To argue correctness, we need to show that decryption works for all layers less than or equal to the permissible depth, L. Assume decryption at depth $d \leq L$, meaning $\|e^T\| \leq nm(nl+1)^d$. Then $s^T \times \mathbf{C} \times G^{-1}(w) = (bs^T\mathbf{G} + e^T)G^{-1}(w) = bs^T\mathbf{G}G^{-1}(w) + e^TG^{-1}(w) = b\langle s^T, w\rangle + \langle e^T, G^{-1}(w)\rangle = b\frac{q}{2} + \langle e^T, G^{-1}(w)\rangle$ where $|\langle e^T, G^{-1}(w)\rangle| \leq \|e^T\| \leq nm(nl+1)^d$. To ensure the scheme can handle circuits of depth L, the parameters must satisfy $nm(nl + 1)^L < \frac{q}{4}$. In other words, the global modulus q must be larger for greater depth parameter L. For explicit parameters n, q, χ, m that satisfies the above condition while maintaining LWE hardness assumption, see [Hal17].

The security of the scheme follows from the LWE hardness assumption; since $\mathbf{A}'$ is pseudorandom and $\mathbf{R}$ is a uniformly random matrix over $\{0, 1\}$, the $n \times nl$ matrix $\mathbf{A}'\mathbf{R}$ simply contains nl LWE samples. Therefore, adding $b\mathbf{G}$ is computationally indistinguishable, meaning the ciphertext C is pseudorandom.

Notice that since q is dependent on maximum depth L, the GSW scheme is leveled and not pure FHE as it currently stands. However, the bootstrapping theorem together with circular security assumption can be used to turn the leveled GSW scheme into a pure FHE scheme. To decrypt a ciphertext homomorphically, we first calculate $v \overset{\text{def}}{=} \mathbf{C} \times G^{-1}(w)$ using only public information as a pre-processing step. Then, the actual decryption is done by computing $[\langle s^T, v\rangle]_q$. It turns out that modular inner products can be calculated by circuits with depth logarithmic in the size of the inputs, meaning depth $O(\log(s+v)) = O(\log(n \log q)) = O(\log n + \log\log(q))$.[1]

Theorem 8. *The GSW scheme is bootstrappable.*

Proof. We want to show that the augmented decryption function can be evaluated homomorphically for some set of parameters. Assume augmented decryption function has depth $d = O(\log(n) + \log\log(q))$ for some real number. Then, the parameters must satisfy $4nm(nl+1)^d < q$. Since $m, n = \text{poly}(\lambda)$, there exists some integer γ such that $4nm(nl+1)^d \leq (nl)^{\gamma+d} = (n\lceil \log q\rceil)^{\gamma+d} = (n\lceil \log q\rceil)^{O(\log(n)+\log\log(q))}$. Therefore, it is sufficient to show that there exists parameters such that $(n\lceil \log q\rceil)^{\delta(\log(n)+\log\log(q))} \leq q$ for some real value δ and sufficiently large λ. Let $q = 2^{3\delta\log^2(n)}$.

$$\begin{aligned}
(n\lceil \log q\rceil)^{\delta(\log n+\log\log q)} &= (n\lceil 3\delta\log^2(n)\rceil)^{\delta(\log n+\log^2 q)} \\
&\leq \left(2^{\log(n)}\right)^{\delta(\log n+\log^2 q)} \\
&= \left(2^{\log(n)}\right)^{\delta(\log n+\log(3\delta\log^2 n))} \\
&\leq \left(2^{\log(n)}\right)^{\delta(\log n+\log(3\delta)+\log^3 n)} \\
&\leq \left(2^{\log(n)}\right)^{\delta(3\log n)} \\
&= \left(2^{3\delta\log^2(n)}\right) \\
&= q
\end{aligned}$$

Since GSW is bootstrappable, it is pure FHE (under circular security) as per the bootstrapping theorem (see theorem 7). □

8.2 Efficiency improvements

[1]To see how to construct such a circuit, see [Hal17].

Bibliography

[ABD16] Albrecht, Martin, Bai, Shi, and Ducas, Léo. *A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and Graded Encoding Schemes.* Cryptology ePrint Archive, Paper 2016/127. `https://eprint.iacr.org/2016/127`. 2016. URL: `https://eprint.iacr.org/2016/127`.

[AP13] Alperin-Sheriff, Jacob and Peikert, Chris. *Practical Bootstrapping in Quasilinear Time.* Cryptology ePrint Archive, Paper 2013/372. `https://eprint.iacr.org/2013/372`. 2013. URL: `https://eprint.iacr.org/2013/372`.

[App+09] Applebaum, Benny, Cash, David, Peikert, Chris, and Sahai, Amit. “Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems”. In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference.* Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 595–618. DOI: `10.1007/978-3-642-03356-8_35`. URL: `https://www.iacr.org/archive/crypto2009/56770585/56770585.pdf`.

[AB09] Arora, Sanjeev and Barak, Boaz. *Computational Complexity: A Modern Approach.* 1st. New York, NY, USA: Cambridge University Press, 2009.

[Bra12] Brakerski, Zvika. *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP.* Cryptology ePrint Archive, Paper 2012/078. `https://eprint.iacr.org/2012/078`. 2012. URL: `https://eprint.iacr.org/2012/078`.

[Bra18] Brakerski, Zvika. “Fundamentals of Fully Homomorphic Encryption - A Survey”. In: *Electronic Colloquium on Computational Complexity* 125 (2018).

[BGV14] Brakerski, Zvika, Gentry, Craig, and Vaikuntanathan, Vinod. “(Leveled) Fully Homomorphic Encryption Without Bootstrapping”. In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014). Preliminary version in ITCS 2012, p. 13.

[Bra+13] Brakerski, Zvika, Langlois, Adeline, Peikert, Chris, Regev, Oded, and Stehlé, Damien. *Classical Hardness of Learning with Errors.* 2013. arXiv: `1306.0281 [cs.CC]`.

[BV11] Brakerski, Zvika and Vaikuntanathan, Vinod. *Efficient Fully Homomorphic Encryption from (Standard) LWE.* Cryptology ePrint Archive, Paper 2011/344. `https://eprint.iacr.org/2011/344`. 2011. URL: `https://eprint.iacr.org/2011/344`.

[Che+18] Cheon, Jung Hee, Han, Kyoohyung, Kim, Andrey, Kim, Miran, and Song, Yongsoo. *Bootstrapping for Approximate Homomorphic Encryption.* Cryptology ePrint Archive, Paper 2018/153. `https://eprint.iacr.org/2018/153`. 2018. URL: `https://eprint.iacr.org/2018/153`.

[Che+16] Cheon, Jung Hee, Kim, Andrey, Kim, Miran, and Song, Yongsoo. *Homomorphic Encryption for Arithmetic of Approximate Numbers.* Cryptology ePrint Archive, Paper 2016/421. `https://eprint.iacr.org/2016/421`. 2016. URL: `https://eprint.iacr.org/2016/421`.

[Chi+18] Chillotti, Ilaria, Gama, Nicolas, Georgieva, Mariya, and Izabachène, Malika. *TFHE: Fast Fully Homomorphic Encryption over the Torus.* Cryptology ePrint Archive, Paper 2018/421. `https://eprint.iacr.org/2018/421`. 2018. URL: `https://eprint.iacr.org/2018/421`.

[CNT11] Coron, Jean-Sebastien, Naccache, David, and Tibouchi, Mehdi. *Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers.* Cryptology ePrint Archive, Paper 2011/440. `https://eprint.iacr.org/2011/440`. 2011. URL: `https://eprint.iacr.org/2011/440`.

[CLT13] Coron, Jean-Sébastien, Lepoint, Tancrède, and Tibouchi, Mehdi. *Batch Fully Homomorphic Encryption over the Integers.* Cryptology ePrint Archive, Paper 2013/036. `https://eprint.iacr.org/2013/036`. 2013. URL: `https://eprint.iacr.org/2013/036`.

[Dij+09] Dijk, Marten van, Gentry, Craig, Halevi, Shai, and Vaikuntanathan, Vinod. *Fully Homomorphic Encryption over the Integers.* Cryptology ePrint Archive, Paper 2009/616. `https://eprint.iacr.org/2009/616`. 2009. URL: `https://eprint.iacr.org/2009/616`.

[DM14] Ducas, Léo and Micciancio, Daniele. *FHEW: Bootstrapping Homomorphic Encryption in less than a second.* Cryptology ePrint Archive, Paper 2014/816. `https://eprint.iacr.org/2014/816`. 2014. URL: `https://eprint.iacr.org/2014/816`.

[FV12] Fan, Junfeng and Vercauteren, Frederik. *Somewhat Practical Fully Homomorphic Encryption.* Cryptology ePrint Archive, Paper 2012/144. `https://eprint.iacr.org/2012/144`. 2012. URL: `https://eprint.iacr.org/2012/144`.

[Gam+14] Gama, Nicolas, Izabachene, Malika, Nguyen, Phong Q., and Xie, Xiang. *Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions and Homomorphic Cryptosystems.* Cryptology ePrint Archive, Paper 2014/283. `https://eprint.iacr.org/2014/283`. 2014. URL: `https://eprint.iacr.org/2014/283`.

[Gar23] Gartner. *Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly $600 Billion in 2023.* Accessed: 2023-06-07. 2023. URL: `https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023`.

[Gen09] Gentry, Craig. "A Fully Homomorphic Encryption Scheme". AAI3382729. PhD thesis. Stanford, CA, USA: Stanford University, 2009.

[GH11] Gentry, Craig and Halevi, Shai. *Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits.* Cryptology ePrint Archive, Paper 2011/279. `https://eprint.iacr.org/2011/279`. 2011. URL: `https://eprint.iacr.org/2011/279`.

[GH10] Gentry, Craig and Halevi, Shai. *Implementing Gentry's Fully-Homomorphic Encryption Scheme.* Cryptology ePrint Archive, Paper 2010/520. `https://eprint.iacr.org/2010/520`. 2010. URL: `https://eprint.iacr.org/2010/520`.

[GSW13] Gentry, Craig, Sahai, Amit, and Waters, Brent. *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based.* Cryptology ePrint Archive, Paper 2013/340. `https://eprint.iacr.org/2013/340`. 2013. URL: `https://eprint.iacr.org/2013/340`.

[Gol08] Goldreich, Oded. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008. DOI: 10.1017/CBO9780511804106.

[Gol01] Goldreich, Oded. *Foundations of Cryptography.* Vol. I. Cambridge University Press, 2001.

[Gol04] Goldreich, Oded. *Foundations of Cryptography.* Vol. II. Cambridge University Press, 2004.

[Hal17] Halevi, Shai. "Homomorphic Encryption". In: *Tutorials on the Foundations of Cryptography.* Ed. by Yehuda Lindell. Information Security and Cryptography. Cham: Springer International Publishing, 2017. Chap. 5, pp. 219–268. ISBN: 978-3-319-57047-1. DOI: 10.1007/978-3-319-57048-8. URL: https://doi.org/10.1007/978-3-319-57048-8.

[HPS98] Hoffstein, Jeffrey, Pipher, Jill, and Silverman, Joseph H. "NTRU: A ring-based public key cryptosystem". In: *Algorithmic Number Theory.* Ed. by Joe P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288. ISBN: 978-3-540-69113-6.

[Kal21] Kallenberg, Olav. *Probability Theory and Stochastic Modelling.* 3rd ed. Probability Theory and Stochastic Modelling. Cham, Switzerland: Springer Nature Switzerland AG, 2021. ISBN: 978-3-030-61870-4. DOI: 10.1007/978-3-030-61871-1.

[Kim+13] Kim, Jinsu, Lee, Moon Sung, Yun, Aaram, and Cheon, Jung Hee. *CRT-based Fully Homomorphic Encryption over the Integers.* Cryptology ePrint Archive, Paper 2013/057. https://eprint.iacr.org/2013/057. 2013. URL: https://eprint.iacr.org/2013/057.

[LM20] Li, Baiyu and Micciancio, Daniele. *On the Security of Homomorphic Encryption on Approximate Numbers.* Cryptology ePrint Archive, Paper 2020/1533. https://eprint.iacr.org/2020/1533. 2020. URL: https://eprint.iacr.org/2020/1533.

[LPR12] Lyubashevsky, Vadim, Peikert, Chris, and Regev, Oded. *On Ideal Lattices and Learning with Errors Over Rings.* Cryptology ePrint Archive, Paper 2012/230. https://eprint.iacr.org/2012/230. 2012. URL: https://eprint.iacr.org/2012/230.

[MP20] Micciancio, Daniele and Polyakov, Yuriy. *Bootstrapping in FHEW-like Cryptosystems*. Cryptology ePrint Archive, Paper 2020/086. https://eprint.iacr.org/2020/086. 2020. URL: https://eprint.iacr.org/2020/086.

[MR07] Micciancio, Daniele and Regev, Oded. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *SIAM Journal on Computing* 37.1 (2007), pp. 267–302. DOI: 10.1137/S0097539705447360. eprint: https://doi.org/10.1137/S0097539705447360. URL: https://doi.org/10.1137/S0097539705447360.

[MF21] Mittelbach, Arno and Fischlin, Marc. *The Theory of Hash Functions and Random Oracles*. Vol. I. Springer Cham, 2021.

[Pei15] Peikert, Chris. *A Decade of Lattice Cryptography*. Cryptology ePrint Archive, Paper 2015/939. https://eprint.iacr.org/2015/939. 2015. URL: https://eprint.iacr.org/2015/939.

[Pei08] Peikert, Chris. *Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem*. Cryptology ePrint Archive, Paper 2008/481. https://eprint.iacr.org/2008/481. 2008. URL: https://eprint.iacr.org/2008/481.

[Reg05] Regev, Oded. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[RAD78] Rivest, Ron, Adleman, Leonard, and Dertouzos, Michael. "On data banks and privacy homomorphisms". In: *Found. of Sec. Comp.* 1978, pp. 169–180.

[SS11] Scholl, P. and Smart, N. P. *Improved Key Generation For Gentry's Fully Homomorphic Encryption Scheme*. Cryptology ePrint Archive, Paper 2011/471. https://eprint.iacr.org/2011/471. 2011. URL: https://eprint.iacr.org/2011/471.

[SV09] Smart, N. P. and Vercauteren, F. *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*. Cryptology ePrint Archive, Paper 2009/571. https://eprint.iacr.org/2009/571. 2009. URL: https://eprint.iacr.org/2009/571.

[Vai11] Vaikuntanathan, Vinod. “Computing Blindfolded: New Developments in Fully Homomorphic Encryption”. In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. University of Toronto. IEEE, 2011. DOI: 10.1109/FOCS.2011.98.

[Ver11] Vershynin, Roman. *Introduction to the non-asymptotic analysis of random matrices*. 2011. arXiv: 1011.3027 [math.PR].

*

A. Appendix

Below is the code for Figure 1.

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import math

def sample_discrete_gaussian(sigma, size=1):
    """Sample from the discrete Gaussian distribution."""
    x = np.arange(-3*sigma, 3*sigma+1)
    pmf = np.exp(-x**2 / (2 * sigma**2))
    pmf /= pmf.sum()
    return np.random.choice(x, size=size, p=pmf)

# Set parameters
n_samples = 1000000
alpha = 100
sigma = alpha / math.sqrt(2 * math.pi)

# Generate discrete Gaussian samples in Z^2
samples = np.array([sample_discrete_gaussian(sigma, size=n_samples) for _
    in range(2)]).T

# Compute 2D histogram
H, xedges, yedges = np.histogram2d(samples[:, 0], samples[:, 1], bins=(30,
    30))

# Get x, y, z for non-zero counts
x, y = np.nonzero(H)
z = H[x, y]
x = xedges[x]
y = yedges[y]

# Create a new 3D subplot
fig = plt.figure(figsize=(8, 6))  # Define the figure size
ax = fig.add_subplot(111, projection='3d')
```

```

# Plot a dot at the top of each line
scatter = ax.scatter(x, y, z, color='black', s=6, alpha=1)

# Set labels with larger font size
ax.set_zlabel('Frequency', fontsize=14)

# Set grid color to white and linestyle to dotted
ax.grid(color='white', linestyle=':', linewidth=0.75)

# Set the background color of the 3d plot to be transparent
ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

# Make the grid lines transparent
ax.xaxis.pane.set_edgecolor('w')
ax.yaxis.pane.set_edgecolor('w')
ax.zaxis.pane.set_edgecolor('w')

ax.grid(True)
plt.tight_layout()
plt.show()
```