TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

T H E S I S

The development of homomorphic cryptography

From RSA to Gentry's privacy homomorphism

presented to the Institute of
Discrete Mathematics and Geometry
Vienna University of Technology

Supervisor
Univ.Prof. Dipl.-Ing. Dr.techn. Michael Drmota

by
Sigrun Goluch
Taborstraße 71/3
1020 Wien

Date

Signature

Contents

1. Introduction

Ever since the discovery of public-key cryptography by Diffie and Hellman [12] in the year 1976, the necessity for total privacy of digital data has become stronger and stronger, especially since the internet has become an indispensable part of both our private and work lives. Naturally, the question for more and more secure encryption schemes arose in the past few decades. One way to achieve confidentiality in applications, such as online banking, electronic voting, virtual networks etc. are homomorphic and especially fully homomorphic cryptographic schemes.

Fully homomorphic cryptosystems or privacy homomorphisms were introduced by Rivest, Adleman, and Dertouzous in 1978 [37]. In their paper they asked for a way to allow a third, untrusted party to carry out extensive computation on encrypted data, without having to decrypt first. Unfortunately, shortly after its publication, major security flaws were found in the original proposed schemes of Rivest et al. The search for fully homomorphic cryptosystems began.

The aim of homomorphic cryptography is to ensure privacy of data in communication and storage processes, such as the ability to delegate computations to untrusted parties. If a user could take a problem defined in one algebraic system and encode it into a problem in a different algebraic system in a way that decoding back to the original algebraic system is hard, then the user could encode expensive computations and send them to the untrusted party. This untrusted party then performs the corresponding computation in the second algebraic system, returning the result to the user. Upon receiving the result, the user can decode it into a solution in the original algebraic system, while the untrusted party learns nothing of which computation was actually performed.

Over the years a lot of either additively (Paillier [35] 1999, Goldwasser-Micali [23] 1984, Naccache-Stern [34] 1998) or multiplicatively (El Gamal [14] 1984, RSA [37] 1978) homomorphic schemes have been introduced to the world. The demand for a fully homomorphic cryptosystem rose again in 1991 when Feigenbaum et al. [15] asked: "*Is there an encryption function* $\mathsf{Enc}()$ *such that both* $\mathsf{Enc}(x+y)$ *and* $\mathsf{Enc}(x \cdot y)$ *are easy to compute from* $\mathsf{Enc}(x)$ *and* $\mathsf{Enc}(y)$*?*" and was answered in 2009. Craig Gentry published his fully homomorphic cryptosystem [19] in the summer of 2009. Although not yet useful for practical applications, it ended the long search for the in 1978 emerged question about the existence of *privacy homomorphisms*.

This thesis provides an overview of three cryptographic systems, starting with the

first ever published partial homomorphic encryption scheme. This is followed by an additively homomorphic one and ends with the first ever published fully homomorphic system. Prior to that of course the needed mathematic background necessary to understand the schemes.

Chapter 2 presents an overview of the schemes underlying mathematic foundations, such as groups, rings, primes and lattices. It is split into three sections: algebraic and number theoretic foundation (Section 2.1), complexity theory (Section 2.2) and a short introduction of measure theoretic probability (Section 2.3).

The central subject of chapter 3 are public key cryptosystems, which build the computational foundation for this thesis, and the consequential notions of security.

The following chapter 4 provides the definition for homomorphic public key cryptosystems and especially fully homomorphic schemes, as well as some examples of the range of application for homomorphic cryptography.

Chapters 5, 6 and 7 each describe a (fully) homomorphic encryption scheme in detail.
Starting with RSA (chapter 5) since this was actually the first discovered homomorphic scheme, representing the branch of multiplicatively homomorphic cryptosystem.
The second scheme is Paillier's cryptosystem (chapter 6). It is a homomorphic encryption scheme with respect to addition, which can be used to realize such things as electronic voting. An example of this usage can be found in section 4.2.
The last scheme is the most compelling one. Gentry's scheme (chapter 7) is the first ever published fully homomorphic cryptosystem, meaning under both multiplication and addition, and opened the doors for an ample amount of fully homomorphic schemes which followed Gentry's progress. They all have one thing in common, they use something called *bootstrapping* to achieve the state of an fully homomorphic encryption scheme. Unfortunately for now the system is far too extensive to be used in realistic applications.

> [...] *Visions of a fully homomorphic cryptosystem have been dancing in cryptographers' heads for thirty years. I never expected to see one. It will be years before a sufficient number of cryptographers examine the algorithm that we can have any confidence that the scheme is secure, but – practicality be damned – this is an amazing piece of work.* (Bruce Schneier)

2. Preliminaries

This chapter is a collection of basic mathematical material from the mathematical fields *number theory* and *abstract algebra* as well as some fundamental statements on computational theory, especially complexity. The therein contained definitions and theorems will be used throughout this thesis. As the main focus of this thesis lies on the description of homomorphic public key cryptosystems, this chapter states the most important definitions for the comprehension of the schemes and definitions described in the chapters 3 to 7. For a more elaborate study of cryptographic foundations see the referenced publications.

2.1. Algebraic and Number-Theoretic Foundation

This section will give an algebraic and number theoretic basis for the homomorphic cryptosystems described in chapter 5, 6 and 7. The needed material is taken from the standard works [9, 13, 17, 24, 30]. If not stated otherwise, most theorems and proofs can be found there. The following definitions are well known and basic constructs for mathematicians. For the sake of completeness they shall be outlined in this section.

2.1.1. Groups

Definition 2.1. A **group** $(G, *)$ is a nonempty set G together with a binary operation $* : G \times G \to G,\ (a, b) \mapsto a * b$ such that the following conditions hold:

- *Associativity*:
$$\forall\ a, b, c \in G :\ a * (b * c) = (a * b) * c$$
- Existence of an *identity element* e:
$$\exists\ e \in G\ \forall\ a \in G : a * e = e * a = a$$
- Existence of an *inverse element* a':
$$\forall\ a \in G\ \exists\ a' \in G : a * a' = a' * a = e$$

If the operation $a * b$ is **commutative**, that is, if $a * b = b * a$ then the group is said to be **abelian**.

A **finite** group G is a group which (as a set) is finite. The **order** of a finite group (denoted as $|G|$) is the number of elements in it.
Let $a \in G$. If there exists a positive integer n so that $a^n = e$, then a is said to have **finite order**, and the smallest such positive integer n is called the **order of a**, denoted $|a| = n$..

Subgroups are subsets of groups which are groups of *their own right*, in the following sense:
A subset H of a group G is said to be a **subgroup** if, with the same operation $(*)$ and identity element (e), $(H, *)$ is a group.

Definition 2.2. Let G be a group, and let a be any element of G. The set
$\langle a \rangle = \{x \in G \mid x = a^n \text{ for some } n \in \mathbb{Z}\}$ is called a **cyclic subgroup** generated by a.
The group G is called **cyclic group** if there exists an element $a \in G$ so that $G = \langle a \rangle$. In this case a is called a **generator** of G.

Theorem 2.3 (Lagrange)**.** Let G be a finite group. Let H be a subgroup of G. Then the order of H *divides* the order of G.
Particularly, the order of an element $x \in G$ divides the order of G itself.

Proof. The **left coset** of H by $g \in G$ is per definition

$$gH = \{gh \ : \ h \in H\}$$

Likewise, the **right coset** of H by g is Hg. Then the collection of all left cosets of H forms a *partition* of G, which means that every element of G lies in some left coset of H and the all the left cosets are pairwise disjoint. The first statement obviously holds because $x = x \cdot e \in xH$. As for the second, suppose $xH \cap yH \neq \emptyset$ for $x, y \in G$. Then there exists some $h_1, h_2 \in H$ with $xh_1 = yh_2$ multiplying both sides with h_2^{-1} yields

$$\begin{aligned}(xh_1)h_2^{-1} &= (yh_2)h_2^{-1} \\ &= y(h_2h_2^{-1}) = y \cdot e = y\end{aligned}$$

Since H is a group itself $h_1h_2^{-1} \in H$. Then with $h_1h_2^{-1} = z$:

$$yH = \{yh : \ h \in H\} = \{(xz)h \ : \ h \in H\} = \{x(zh) \ : \ h \in H\}$$

Thus, $yH = xH$, since the relationship between x and y is symmetrical. Therefore the left cosets of H in G form a partition of G.

The next step is to show that the order of the left cosets are identical, by demonstrating a *bijection* from H to xH for any $x \in G$. Now, define the map

$$f: \ H \to xH$$
$$g \mapsto xg$$

If $f(g) = f(g')$, then by definition $xg = xg'$, multiplying both sides with x^{-1} yields $g = g'$. What is left to be shown is the surjectivity of the above map. That is again directly deduced from the definition of f, since $f(h) = xh$. Thus, all the left cosets of H have the same cardinality as H itself.
Since G is the disjoint union of the left cosets of H, $|H|$ divides $|G|$. □

2.1.2. Homomorphisms

Group homomorphisms are the maps of interest among groups and the basis of homomorphic cryptosystems.

Definition 2.4. A function $f : G \longrightarrow H$ from one group G to another H is a **(group) homomorphism** if the group operation is preserved in the sense that

$$f(g_1 *_G g_2) = f(g_1) *_H f(g_2)$$

for all $g_1, g_2 \in G$. Let e_G be the identity in G and e_H the identity in H. A group homomorphism f maps e_G to e_H: $f(e_G) = f(e_H)$.
Note that f must preserve the inverse map due to:

$$f(g)f(g^{-1}) = f(gg^{-1}) = f(e_G) \text{ , therefore: } f(g)^{-1} = f(g^{-1}).$$

The **kernel** of a homomorphism f is

$$\ker f = \{g \in G \ : \ f(g) = e_h\}$$

The **image** of f is like the image of any function

$$\text{im } f = \{h \in H \ : \ \exists g \in G \text{ such that } f(g) = h\}$$

If a group homomorphism $f : G \to H$ is *surjective*, then H is said to be a **homomorphic image** of G.
If a group homomorphism $f : G \to H$ has an *inverse* homomorphism, then f is said to be an **isomorphism**, and G and H are said to be **isomorphic**, denoted as

$$G \cong H$$

If a group homomorphism is a *bijection*, then it has an inverse which is a group homomorphism, so it is an *isomorphism*.

Example (Exponential functions for groups)**.** Let G be any group, and let a be any element of G. Define $f : \mathbb{Z} \longrightarrow G$ by $f(n) = a^n$, for all $n \in \mathbb{Z}$. This is a group homomorphism from $\mathbb{Z}$ to G.

2.1.3. Congruences

For the study of almost any public key cryptosystem the theory of congruences is of crucial importance. Recall that a *relation* R on a set S is a subset of the Cartesian product $S \times S$, denoted as $x\ R\ y$, if the ordered pair (x, y) lies in the subset R of $S \times S$. Now:

Definition 2.5. Let A be a set and $\sim$ be a binary relation on A. $\sim$ is called an **equivalence relation** if and only if for all $a, b, c \in A$, all the following holds true:

1. *Reflexivity:* $a \sim a$
2. *Symmetry:* if $a \sim b$ then $b \sim a$
3. *Transitivity:* if $a \sim b$ and $b \sim c$ then $a \sim c$

The **equivalence class** of a under $\sim$, denoted $[a]$ or $\overline{a}$, is defined as $\overline{a} = \{b \in A \mid a \sim b\}$. Because of the $reflexivity$ it holds that a is in $\overline{a}$. It also holds that any two equivalence classes, $\overline{a}, \overline{b}$ are either *equal* or *disjoint*. It follows that the *set of all equivalence classes* of A (or any other basic set) forms a **partition** of A.

Definition 2.6. Let n be a fixed positive integer. Two integers a and b are said to be **congruent modulo n** symbolized by $a \equiv b \pmod{n}$, if $n|(a - b)$; that is, provided that $a - b = kn$ for some integer k.

Proposition 2.7. *For a fixed modulus n equality modulo n is an equivalence relation:*

1. *Reflexivity:* $a \equiv a \bmod n$
2. *Symmetry: if* $a \equiv b \bmod n$ *then* $b \equiv a \bmod n$
3. *Transitivity: if* $a \equiv b \bmod n$ *and* $b \equiv c \bmod n$ *then* $a \equiv c \bmod n$

The **congruence class** or **residue class** of an integer $x \bmod n$ denoted $\overline{x}$, is the set of all integers equal to $x \bmod n$:

$$\overline{x} = \{y \in \mathbb{Z} \mid y \equiv x \bmod n\} = x + n\mathbb{Z}$$

The **integers mod n**, denoted $\mathbb{Z}_n$, is the set of all congruence classes modulo n.

$$\mathbb{Z}_n = \{\overline{0}, ..., \overline{n-1}\}$$

for $n \neq 0$ then $|\mathbb{Z}_n| = n$.

Corollary 2.8. *Congruences inherit many basic properties from ordinary arithmetics, simply because $x = y$ implies $x = y \bmod n$:*

- *Distributivity:* $x(y + z) \bmod n = xy + xz \bmod n$
- *Associativity of addition:* $(x + y) + z \bmod n = x + (y + z) \bmod n$
- *Associativity of multiplication:* $(xy)z \bmod n = x(yz) \bmod n$
- *Property of 1:* $1 \cdot x \bmod n = x \cdot 1 \bmod n = x \bmod n$

- *Property of 0:* $0 + x \bmod n = x + 0 \bmod n = x \bmod n$

Definition 2.9. The **greatest common divisor** of two non-zero integers, written as $\gcd(\mathbf{a}, \mathbf{b})$, is the largest positive integer that divides a and b without remainder. The **least common multiple** of two integers, written as $\mathrm{lcm}(\mathbf{a}, \mathbf{b})$ is the smallest positive integer that is a multiple of both.

With the above stated properties it is possible to define addition and multiplication on congruence classes as follows:

$$\overline{a} + \overline{b} = \overline{a + b},\ \overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

In this context there are two special cases, taken from group theory.
A **multiplicative inverse mod n** to an integer a is an integer b (if it exists) so that

$$a \cdot b = 1 \bmod n$$

The **multiplicative order mod n** to an integer a is the smallest positive integer k so that

$$a^k = 1 \bmod n$$

it is denoted $ord_n(a)$.

Proposition 2.10. *An integer a has a multiplicative inverse modulo n, if and only if* $\gcd(a, n) = 1$

Proof. From $\gcd(a, n) = 1$ follows that here are integers r, s so that $ra + sn = 1$, and

$$ra = 1 - sn = 1 \bmod n$$

The other implication is trivial. □

It makes sense to define a specified notation for the elements of $\mathbb{Z}_n$ which are invertible for reasons which will become obvious later on. Thus,

$$\mathbb{Z}_n^* = \{\overline{x} \in \mathbb{Z}_n : \gcd(x, n) = 1\}$$

The set $\mathbb{Z}_n^*$ satisfies all group requirements under the ring-multiplication, $(\mathbb{Z}_n^*, \cdot)$(see Def. 2.1) and is therefore called the **multiplicative group** or **group of units** of $\mathbb{Z}_n$.

Definition 2.11. For a positive integer n, the **Euler totient-function** $\varphi(n)$ is the number of integers b so that $1 \leq b \leq n$ and $\gcd(b, n) = 1$. For example, $\varphi(1) = 1, \varphi(2) = 1, \varphi(3) = 2$ and $\varphi(4) = 2$. Note that

$$\varphi(n) = |\mathbb{Z}_n^*|$$

2.1.4. Primes

As prime numbers are an essential topic concerning almost every cryptographic topic, this section gives a short introduction to prime numbers. There are countless numbers of books which specialize in prime number theory, e.g. [9, 17], they are referred for further information.

Definition 2.12. An integer $p > 1$ is called a **prime number** if its only positive divisors are 1 and p

There is a very interesting theorem concerning the existence of prime numbers in a special interval. This turns out to be useful when searching for prime number factors of a composite number.

Theorem 2.13. A positive integer n is prime, if and only if it is not divisible by any of the integers d with $1 < d \leq \sqrt{n}$

Proof. Suppose n has a proper factorization $n = d \cdot e$, where $d \leq e$. Then

$$d = \frac{n}{e} \leq \frac{n}{d}$$

hence $d^2 \leq n$, so $d \leq \sqrt{n}$. □

The previous theorem suggests a procedure for primality testing called **trial division**. Instead of trying to divide a composite number n by all integers $d = 1, 2, ..., n$ it suffices to try only those in the range of $d = 1, 2, ..., \sqrt{n}$.
The following theorem shows the importance of prime numbers in the mathematical world.

Theorem 2.14 (Fundamental Theorem of Arithmetic)**.** Every positive integer n can be expressed as a unique product of primes, apart from the order in which the factors occur.

$$n = p_1^{k_1} p_2^{a_2} ... p_i^{k_i}$$

where for $i = 1, 2..., r$, each k_i is a positive integer and each p_i is a prime, with $p_1 < p_2 < ... < p_k$.

There are multiple ways of proving the correctness of this theorem. It is up to the reader to study those in literature (e.g. [9, 17]), referenced in the introduction of this chapter.

Definition 2.15. Two integers a and b are said to be **coprime** or **relatively prime**, if they have no common positive factor other than 1 or, equivalently, if $\gcd(a, b) = 1$.

Theorem 2.16. A prime p divides a product $a \cdot b$ if and only if either $p|a$ or $p|b$.

Proof. If $p|a$ the proof is complete, otherwise since p is prime, and $\gcd(p, a) \neq p$, it must be that $\gcd(p, a) = 1$. Then let r, s be integers so that $1 = rp + sa$ and let $ab = kp$. Then

$$b = b \cdot 1 = b(rp + sa) = p \cdot (rb + sk)$$

so b is a multiple of p. □

2.1.5. Fermat's little and Euler's theorem

This section provides two very significant theorems of the field *cryptography*. First Fermat's little theorem which was first stated 370 years ago in 1640 by Pierre de Fermat will be presented. Its first published proof by Leonard Euler can be found in the 1736 published paper "*Theorematum Quorundam ad Numeros Primos Spectantium Demonstratio*". Euler himself found a generalization of Fermat's theorem in the same year. Fermat's little theorem is the basis of the self-referential primality test : **Fermat primality test**.

Theorem 2.17 (Fermat's little theorem)**.** For any prime p, and any integer $a \not\equiv 0 \bmod p$, we have $a^{p-1} \equiv 1 \bmod p$. Moreover, for any integer a, we have $a^p \equiv a \bmod p$.

Proof. It is easily shown that the set $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, with the operation of multiplication forms a group. Assume that $1 \leq a \leq p-1$, $k = |a|$, so that $a^k \equiv 1 \bmod p$. Using Lagrange's theorem (2.3), k divides the order of $\mathbb{Z}_p^*$, which is $p-1$. Then

$$a^{p-1} \equiv a^{km} \equiv (a^k)^m \equiv 1^m \equiv 1 \bmod p$$

□

The next theorem is very useful in various ways. It will later be used to prove the well known Euler theorem (see Theorem 2.20). It is also very useful when solving a system of equations. In computer science it is used to make time-consuming computations much faster.

Theorem 2.18 (Chinese remainder theorem)**.** Let $n_1, ..., n_k$ be pairwise relatively prime, positive integers, and let $a_1, ..., a_k$ be arbitrary integers. Then there exists an integer z so that

$$z \equiv a_i \bmod n_i \ (i = 1, ..., k)$$

Moreover, any other integer z' is also a solution of these congruences, if and only if $z \equiv z' \bmod n$, where $n := \prod_{i=1}^{k} n_i$.

Proof. Let $n := \prod_{i=1}^{k} n_i$ and $n_i' := \frac{n}{n_i}$ $(i = 1, ..., k)$. Due to the fact that the n_i's are pairwise prime, it follows that $\gcd(n_i, n_i') = 1$ for $i = 1, ..., k$. Therefore, let

$$m_i := (n_i')^{-1} \bmod n_i \text{ and } w_i := n_i' m_i \ (i = 1, ..., k)$$

By construction it holds $w_i \equiv 1 \bmod n_i$ for $i = 1, ..., k$ and $w_i \equiv 0 \bmod n_j$ for $j = 1, ..., k$ and $j \neq i$. Therefore $w_i \equiv \delta_{ij} \bmod n_j$
with $\delta_{ij} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$
Now, let $z := \sum_{i=1}^{k} w_i a_i$. Obviously

$$z \equiv \sum_{i=1}^{k} w_i a_i \equiv \sum_{i=1}^{k} \delta_{ij} a_i \equiv a_j \bmod n_j \text{ for } j = 1, ..., k.$$

Then z is the solution of the given system of congruences.
Moreover, if $z' \equiv z \bmod n$, then since $n_i | n$ for $i = 1, ..., k$, we see that $z' \equiv z \equiv a_i \bmod n_i$ for $i = 1, ..., k$, then z' also solves the system of congruences.
Finally, if z' solves the system, then $z' \equiv z \bmod n_i$ for $i = 1, ..., k$. That is, $n_i | (z' - z)$ for $i = 1, ..., k$. Since $n_1, ..., n_k$ are pairwise relatively prime, this implies that $n | (z' - z)$, or equivalently, $z' \equiv z \bmod n$. □

Remark. In many books we find the Chinese remainder theorem in the following form:
Let n be the product of two relatively prime integers p and q, then $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$ and $\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$.
Moreover, let f be the function that maps elements $x \in 0, ..., n-1$ to pairs (x_p, x_q) with $x_p \in 0, ..., p-1$ and $x_q \in 0, ..., q-1$ defined by $f(x) = ([x \bmod p], [x \bmod q])$. Then f is an isomorphism from $\mathbb{Z}_n$ to $\mathbb{Z}_p \times \mathbb{Z}_q$ as well as an isomorphism from $\mathbb{Z}_n^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$

The Chinese remainder theorem is very useful when computing products (or even exponentiations) modulo n, when n is the product of distinct primes p, q. It shows that multiplication and addition can be "transformed" to analogous operations modulo p and q, thus making those computations less time consuming. For better understanding see the next example.

Example. We wish to compute the product $11 \cdot 8 \bmod 15$ in $\mathbb{Z}_{15}^*$. As a result of the Chinese remainder theorem it holds that $\mathbb{Z}_{15}^* \cong \mathbb{Z}_5^* \times \mathbb{Z}_3^*$ We map $11 \leftrightarrow (1, 2)$, since $11 \bmod 5 = 1$ and $11 \bmod 3 = 2$, respectively $8 \leftrightarrow (3, 2)$. Then,

$$[11 \cdot 8 \bmod 15] \leftrightarrow (1, 2) \cdot (3, 2) = ([1 \cdot 3 \bmod 5], [2 \cdot 2 \bmod 3]) = (3, 1).$$

Now, it is easy to show that $(3, 1) \leftrightarrow 13$, which is the solution of $11 \cdot 8 = 13 \bmod 15$.

Before stating the next theorem, some useful properties of $\varphi(n)$ are listed.

Corollary 2.19. *Let $\varphi(n)$ be the Euler totient function and n be an integer. Then $\varphi(n)$ has the following properties:*

1. *φ is a multiplicative function: if m and n are coprime then $\varphi(mn) = \varphi(m)\varphi(n)$.*
2. *For p is prime and $k \geq 1 : \varphi(p^a) = (p-1)p^{a-1}$.*
3. *$\varphi(n^k) = n^{k-1}\varphi(n)$.*

Proof. **(1)** Consider

$$\rho : \mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n \tag{2.1}$$

$$[a]_{mn} \mapsto ([a]_m, [a]_n) \tag{2.2}$$

First, note that the definition 2.19 of ρ is distinct, since $a \equiv a' \bmod mn$ implies $a \equiv a' \bmod m$ and $a \equiv a' \bmod n$. Second according to the Chinese remainder theorem (2.18), the map ρ is bijective. Moreover, it is easy to see that $\gcd(a, mn) = 1$, if and only if $\gcd(a, m) = 1$ and $\gcd(a, n) = 1$. Therefore, ρ carries $\mathbb{Z}^*_{mn}$ injectively onto $\mathbb{Z}^*_m \times \mathbb{Z}^*_n$. Hence it holds that $|\mathbb{Z}^*_{mn}| = |\mathbb{Z}^*_m \times \mathbb{Z}^*_n|$.

(2) Let $m = p^a$ be a power of a prime, the numbers that have a common factor with m are

$$1 \cdot p, 2 \cdot p, ..., p^{a-1} \cdot p$$

of which there are precisely p^{a-1}. Thus, $\varphi(p^a) = p^a - p^{a-1} = p^{a-1}(p-1)$.

(3) follows directly from (1) and (2) and the *Fundamental Theorem of Arithmetic.* □

The next theorem is a generalization of Fermat's little theorem:

Theorem 2.20 (Euler's theorem)**.** For x relatively prime to a positive integer n ($\gcd(x, n) = 1$),

$$x^{\varphi(n)} = 1 \bmod n$$

The special case that n is a prime is *Fermat's little theorem* 2.17.

Proof. The numbers a which are relatively prime to n form a group under multiplication mod n ($\mathbb{Z}^*_n$). The order of $\mathbb{Z}^*_n$ is $\varphi(n)$. Let $a \in \mathbb{Z}^*_n$, then the order of a must have a multiple equal the size of $\mathbb{Z}^*_n$. $|a| = k$ is the size of the subgroup of $\mathbb{Z}^*_n$ generated by a, and Lagrange's theorem (2.3) states that the size of any subgroup of $\mathbb{Z}^*_n$ divides $\varphi(n)$.
Thus for some integer $m > 0 : m \cdot k = \varphi(n)$. Therefore

$$a^{\varphi(n)} = a^{m \cdot k} = (a^k)^m = 1^m = 1 \bmod n$$

□

Definition 2.21. For a positive integer n, $\lambda(n)$ denotes the least positive integer t such that $x^t \equiv 1 \pmod{n}$ for all integers x with $\gcd(x, n) = 1$. $\lambda(n)$ is the so-called **Carmichael Function**

Again there are some interesting properties to that function [10]:

1. For p is prime: $\lambda(p) = \varphi(p) = (p-1)$
2. Let $n = p_1^{a_1} p_2^{a_2} ... p_k^{a_k}$ be the prime factorization of n, then

$$\lambda(n) = \operatorname{lcm}(\lambda(p_1^{a_1}), \lambda(p_2^{a_2}), ..., \lambda(p_k^{a_k}))$$

$$\text{with: } \lambda(p_i^{a_i}) = \begin{cases} 2^{a_i - 2} & \text{for } p_i = 2, a > 2 \\ (p_i - 1)p_i^{a_i - 1} & \text{otherwise} \end{cases}$$

Theorem 2.22 (Carmichael's Theorem)**.** For two primes $p, q > 2$ and $n = pq$. For each $x \in \mathbb{Z}_{n^2}^*$:

$$x^{\lambda(n)} = 1 \bmod n \tag{2.3}$$

$$x^{n\lambda(n)} = 1 \bmod n^2 \tag{2.4}$$

Proof. Assume that e be the smallest integer so that $x^e = 1 \bmod n$ for all $x \in \mathbb{Z}_{n^2}^*$, then e is a multiple of the element order of all x in $\mathbb{Z}_{n^2}^*$. In equation (2.3) $e = \lambda(n) = \operatorname{lcm}(p-1, q-1)$, due to the fact that the group order of $\mathbb{Z}_n^*$ is per definition

$$|\mathbb{Z}_n^*| = \varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$$

and Lagrange's theorem (2.3) tells us that the order of x is a divisor of the group order $\varphi(n) = (p-1)(q-1)$.
The same argument applies to equation (2.4). The only difference lies in the group order of $\mathbb{Z}_{n^2}^*$ being

$$\left|\mathbb{Z}_{n^2}^*\right| = \varphi(n^2) = n \cdot \varphi(n) = n \cdot (p-1) \cdot (q-1) \tag{2.5}$$

and the fact that the order of a group element $x \in \mathbb{Z}_{n^2}^*$ divides the order in equation (2.5). Hence the exponent is $n \cdot \lambda(n)$. □

2.1.6. Rings

Definition 2.23. Let R be a set on which two binary operations are defined, called *addition* and *multiplication*, and denoted by $+ : G \times G \to G,\ (a, b) \mapsto a + b$ and $\cdot : G \times G \to G,\ (a, b) \mapsto a \cdot b$. Then $(R, +, \cdot)$ is called a **commutative ring** with respect to these operations, if the following properties hold:

- *Associative laws:*

$$\forall a, b, c \in R : a + (b + c) = (a + b) + c$$
$$\forall a, b, c \in R : a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

- *Commutative laws:*

$$\forall a, b \in R : a + b = b + a$$
$$\forall a, b \in R : a \cdot b = b \cdot a$$

- *Distributive laws:*

$$\forall a, b, c \in R : a \cdot (b + c) = a \cdot b + a \cdot c$$
$$(a + b) \cdot c = a \cdot c + b \cdot c$$

- *Additive identity:*

$$\exists 0 \in R \; \forall a \in R : a + 0 = 0 + a = a$$

- *Additive inverses:*

$$\forall a \in R \; \exists (-a) \in R : a + (-a) = (-a) + a = 0$$

The commutative ring R is called a **commutative ring with identity** if it contains an element 1, assumed to be different from 0, so that for all $a \in R$, $a \cdot 1 = 1 \cdot a = a$. In this case, 1 is called a **multiplicative identity element** or **the unit**.
A nonzero element $a \in R$ is called a **zero divisor** if there exists an element $b \neq 0$ so that $a \cdot b = 0$ or $b \cdot a = 0$.
An **integral domain** is a commutative ring with identity so that for any two elements $a, b \in R$, $a \cdot b = 0$ implies either $a = 0$ or $b = 0$.

Example. The integers mod m, denoted $\mathbb{Z}_m$, form a commutative ring with identity.

Definition 2.24. For a ring $(R, +, \cdot)$, let $(R, +)$ be the underlying *additive group*. A subset I is called an **ideal** of R, $I \trianglelefteq R$, if the following conditions hold:

- $(I, +)$ is a subgroup of $(R, +)$
- for all $x \in I$ and for all $r \in R$, $x \cdot r$ and $r \cdot x$ are in I.

The sum and product of two ideals I and J are $\{i + j \;:\; i \in I, j \in J\}$ and the additive closure of $\{i \cdot j \;:\; i \in I, j \in J\}$ respectively.
A **principal ideal** is an ideal $I \trianglelefteq R$ that is generated by a single element a of R, i.e. $(a) = I$.
Two ideals I, J in the commutative ring R are called **coprime** or **relatively prime** if $I + J = R$.

Next, a construction very similar to the factor groups of group theory follows. It is called a quotient ring:

Definition 2.25. Let R be a commutative ring with unit 1. Let I be an ideal in R, $I \trianglelefteq R$. The **quotient ring** R/I is the set of cosets

$$r + I = \{r + i \ : \ i \in I\}$$

with operations of addition and multiplication on R/I defined by

$$(r + I) + (s + I) = = (r + s) + I$$
$$(r + I) \cdot (s + I) = (r \cdot s) + I$$

Example. The zero in the quotient ring is $0_{R/I} = 0+I$, and the unit is $1_{R/I} = 1+I$. Let $R = \mathbb{Z}$ and $I = 8\mathbb{Z}$ (the multiples of 8), then the quotient ring is $\mathbb{Z}_8 = \mathbb{Z}/8\mathbb{Z}$.

Definition 2.26. Let R be a commutative ring. The set of all polynomials

$$c_n x^n + c_{n-1} x^{n-q} + ... + c_1 x + c_0$$

in an indeterminate x with coefficients $c_0, c_1, ..., c_n$ in the ring R themselves constitute a ring, which will be denoted $R[x]$.

Let $p(x) \in R[x]$ be a monic polynomial of degree n

$$p(x) = x^n + p_{n-1} x^{n-1} + ... + p_0.$$

Of special interest will be the ring of all polynomials modulo $p(x)$ denoted as

$$R_p := R[x]/p(x)$$

The following definition concerns a generalization of an ideal, namely a fractional ideal in a number field. Before it is possible to define those fractional ideals, it is necessary to remember some basic structures.

Definition 2.27. Let R be an integral domain. We can define addition and multiplication for fractions $\frac{a}{b} (a, b \in R, b \neq 0)$ as follows

$$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d + b \cdot c}{b \cdot c} \qquad\qquad \frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}.$$

Two fractions $\frac{a}{b}, \frac{c}{d}$ are said to be equivalent if $a \cdot d = b \cdot c$. This equivalence relation is compatible with the operations $+, \cdot$. The quotient set by the above defined equivalence relation is the **field of fractions** of R. It is also the smallest field in which R can be embedded.

Example. The field of fractions of the ring of integers $\mathbb{Z}$ is the field of rationals $\mathbb{Q}$.

Definition 2.28. Let $R \subseteq K$ be an integral domain and K a field. $M \subseteq K$ is called a **R-module**, if following conditions hold:

- $(M, +)$ forms a group,
- for all $a \in M$ and for all $r \in R,\ r \cdot a$ is in M.

Now let R again be an integral domain, but now K be its field of fractions. A R-module $I \neq 0,\ I \subseteq K$ is called a **fractional ideal** if there exists an $a \in R,\ a \neq 0$ such that $a \cdot I \subseteq R$ is an ideal.

Example. For a given $n \in \mathbb{N}$ the set $I := \{\frac{a}{n} | a \in \mathbb{Z}\}$ of all rational numbers with denominator n is a fractional ideal in $\mathbb{Q}$.

Definition 2.29. If R is an integral domain with quotient field K, then the set of all fractional ideals of R forms a commutative monoid, with identity R and multiplication

$$I \cdot J = \left\{ \sum_{i=1}^{n} a_i b_i \mid a_i \in I;\ b_i \in J,\ n \in \mathbb{N} \right\}.$$

A fractional ideal $I \trianglelefteq R$ is said to be **invertible** if $I \cdot J = R$ for some fractional ideal $J \trianglelefteq R$. The inverse of an invertible fractional ideal is unique.

$$I^{-1} = \{a \in K \mid aI \subset R\}.$$

Example. Every principal ideal $I = (a)$ with $a \neq 0$ in an integral domain R is invertible. Let K be the quotient field of R, then $I^{-1} = Rb \subset K$ with $b = 1_R/a$

2.1.7. Lattices

Lattices play a very important role in modern cryptography, especially since the so called post-quantum cryptography is of great interest to the cryptographers of this time. Despite their apparent simplicity, lattices hide a rich combinatorial structure. It is believed that lattice-based cryptography and therefore their underlying *hard* problems are secure against large quantum computer attacks, unlike the factoring problem which was discovered to be solvable in polynomial time on a quantum computer by Shor in 1994 [40].
In this thesis lattices are the fundament of the first ever published fully homomorphic encryption scheme [19]. The basic definitions needed to understand lattice based cryptography are taken from [5, 32], as are the proofs unless stated otherwise.

Notation: In the following, vectors are written in column form using bold lower-case letters, e.g. $\vec{v}$; matrices are written as bold capital letters, e.g., $\mathbf{B}$; $\vec{b_i}$ is the i^{th} column. $\|\vec{v}\|$ denotes the Euclidean length of a vector $\vec{v}$. For matrix $\mathbf{B}$, $\|\mathbf{B}\|$ denotes the length of the longest column vector in $\mathbf{B}$.

Definition 2.30. Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. A **lattice** in $\mathbb{R}^m$ is the set

$$\mathcal{L}(\vec{b}_1, ..., \vec{b}_n) = \left\{ \sum_{i=1}^{n} x_i \vec{b}_i \ : \ x_i \in \mathbb{Z} \right\} = \{\mathbf{B}\vec{x} : \vec{x} \in \mathbb{Z}^n\} \subset \text{span}(\mathbf{B}) = \{\mathbf{B}\vec{x} : \vec{x} \in \mathbb{R}^n\}$$

of all the integer linear combinations of n linearly independent vectors $\{\vec{b}_1, ..., \vec{b}_n\}$ in $\mathbb{R}^m$ $(m \geq n)$. n is called the **rank** and m the **dimension** of the lattice. The sequence of vectors $\{\vec{b}_1, ..., \vec{b}_n\}$ is called a **lattice basis**. It is represented as the following matrix:

$$\mathbf{B} = [\vec{b}_1, ..., \vec{b}_n] \in \mathbb{R}^{m \times n}, \ m \geq n$$

If $n = m$, the lattice is called **full-dimensional** or **full rank**. Note that a lattice has many different bases. Thus, lattices can be characterized without reference to any basis.

Example.

$$\vec{b}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \vec{b}_1^* = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$
$$\vec{b}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \vec{b}_2^* = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

are 2 bases for the 2-dimensional lattice shown in Fig 2.1.

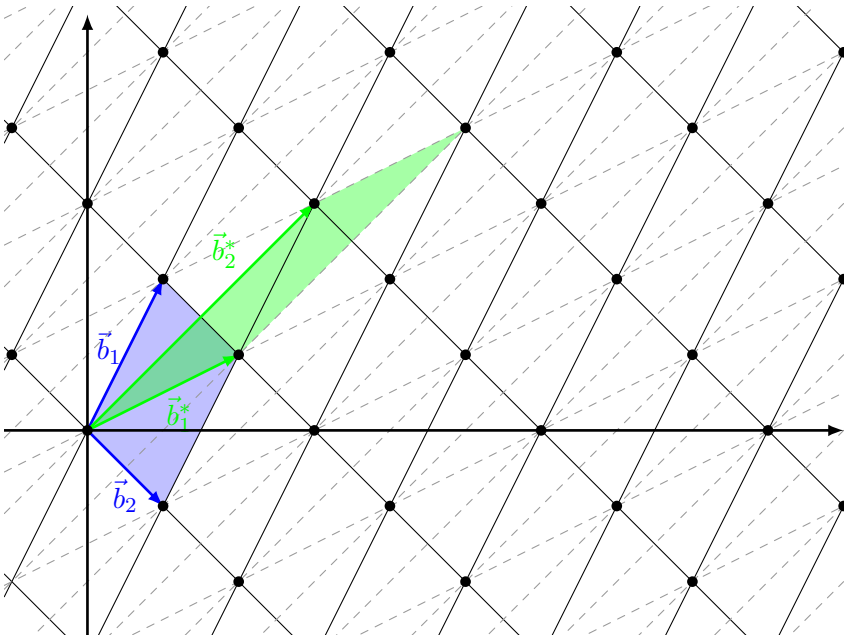

Figure 2.1.: Example of two different bases of the same lattice

A lattice can be defined as a non-empty set Λ of $\mathbb{R}^n$ which is closed under subtraction (if $\vec{x} \in \Lambda$ and $\vec{y} \in \Lambda$, then also $\vec{x} - \vec{y} \in \Lambda$) and discrete (there exists a positive real $\Lambda > 0$ so that the distance between any two lattice vectors is at least Λ). Notice that Λ always contains $\vec{0} = \vec{x} - \vec{x}$ and is closed under complement (if $\vec{x} \in \Lambda$ then $-\vec{x} = \vec{0} - \vec{x} \in \Lambda$) and addition (if $\vec{x}, \vec{y} \in \Lambda$ then $\vec{x} + \vec{y} \in \Lambda$). Therefore, it is an additive subgroup of $\mathbb{R}^n$. In fact, an alternative formulation of the definition of lattice is a discrete additive subgroup of $\mathbb{R}^n$.

Definition 2.31. A basis $\mathbf{B} = (\vec{b}_i, ..., \vec{b}_n) \in \mathbb{Z}^{n \times n}$ is said to be in **Hermite Normal Form** (HNF) if

$$b_{i,j} = \begin{cases} 0 & \text{for } i > j \\ 0 \leq b_{i,j} \leq b_{i,i} & \text{otherwise} \end{cases} \tag{2.6}$$

The HNF of a lattice is unique and can be computed in polynomial time given any basis which makes it a worst-case basis.

An example of n-dimensional lattice is given by the set $\mathbb{Z}^n$ of all integer vectors. A possible basis are the standard unit vectors $\vec{e}_i = [0, .., ., 1, 0, ..., 0]^T$.

It is easy to see that span($\mathbf{B}$) does not depend on a particular basis, i.e., if $\mathbf{B}$ and $\widetilde{\mathbf{B}}$ generate the same lattice then span($\mathbf{B}$) = span($\widetilde{\mathbf{B}}$), so for any lattice $\Lambda = \mathcal{L}(\mathbf{B})$, it is possible to define the linear span, span(Λ) without reference to any basis. The following statements hold:

- $\mathbf{B}$ is a basis of span($\mathbf{B}$) as a vector space.
- The rank of lattice $\mathcal{L}(\mathbf{B})$ equals the dimension of span($\mathbf{B}$) as a vector space over $\mathbb{R}$.
- The rank of lattice $\mathcal{L}(\mathbf{B})$ is a *lattice invariant.*

It is obvious that any set of n linearly independent lattice vectors $\vec{c}_1, ..., \vec{c}_n \in \mathcal{L}(\mathbf{B})$ is a basis of span($\mathbf{B}$), but not necessarily a lattice basis for $\mathcal{L}(\mathbf{B})$.

Example.

$$\vec{c}_1 = \vec{b}_1 + \vec{b}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \vec{c}_2 = \vec{b}_1 - \vec{b}_2 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$\mathbf{C}$ is a basis of the plane $\mathbb{R}^2 = \text{span}(\vec{b}_1, \vec{b}_2)$ since $\vec{c}_1, \vec{c}_2$ are linearly independent. It is not a basis of the lattice $\mathcal{L}(\mathbf{B})$. For this see fig.2.2

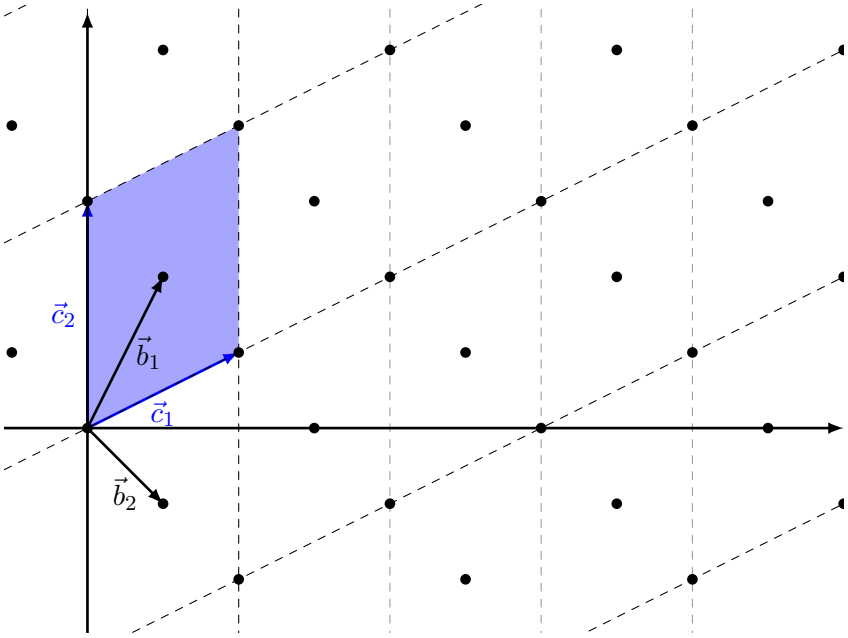

Figure 2.2.: $(\vec{c}_1, \vec{c}_2)$ is not a basis for $\mathcal{L}(\mathbf{B})$

There is a simple geometric characterization for linearly independent lattice vectors that generate the whole lattice:

Definition 2.32. Associated to n linearly independent lattice vectors $\mathbf{C} = [\vec{c}_1, ..., \mathbf{c}_n]$, $\mathbf{c}_i \in \mathcal{L}(\mathbf{B}) \subset \mathbb{R}^m$ for all $i = 1, ..., n$, is the half open **fundamental parallelepiped**

$$\mathcal{P}(\mathbf{C}) = \left\{ \mathbf{C}\vec{x} \ : \ x_i \in \left(-\frac{1}{2}, \frac{1}{2} \right] \right\}.$$

Then, $\mathbf{C}$ is a lattice basis for $\mathcal{L}(\mathbf{B})$, if and only if $\mathcal{P}(\mathbf{C})$ does not contain any lattice vectors other than the origin.

For a vector $\vec{v} \in \mathbb{R}^n$, $\vec{v} \bmod \mathbf{B}$ is the unique vector $\widetilde{\vec{v}} \in \mathcal{P}(\mathbf{B})$ so that $\vec{v} - \widetilde{\vec{v}} \in \mathcal{L}$. Given $\vec{v}$ and $\mathbf{B}$, $\vec{v} \bmod \mathbf{B}$ can be computed efficiently as

$$\vec{v} \bmod \mathbf{B} = \vec{v} - \mathbf{B} \cdot \left\lfloor \mathbf{B}^{-1} \cdot \vec{v} \right\rceil = \mathbf{B} \cdot \left[\mathbf{B}^{-1} \cdot \vec{v} \right],$$

where $\lfloor \cdot \rceil$ rounds the coefficients of a vector to the nearest integers, and $[\cdot]$ denotes the distance between the coefficients of a vector and the nearest integers.

Example. $\vec{q} = \begin{pmatrix} 13/5 \\ 18/7 \\ 7/3 \end{pmatrix}$; $[\vec{q}] = \begin{pmatrix} -2/5 \\ -3/7 \\ 1/3 \end{pmatrix}$; $\lfloor \vec{q} \rceil = \begin{pmatrix} 3 \\ 3 \\ 2 \end{pmatrix}$

Let $\text{dist}(\mathcal{L}, \vec{t}) = \min_{\vec{v}\in\mathcal{L}}\{||\vec{t} - \vec{v}||\}$ denote the *minimum distance* from the target vector $\vec{t}$ to the lattice $\mathcal{L}$. Then for any basis $\mathbf{B}$ is holds that

$$||\vec{t} \bmod \mathbf{B}|| \geq \text{dist}(\mathcal{L}, \vec{t}).$$

Definition 2.33. The **determinant** of a lattice Λ, denoted $\det(\Lambda)$, is the $n-$dimensional volume of the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.
The determinant is a *lattice invariant.*

In the context of lattice based cryptography there are "good" and "bad" bases to a lattice. A Basis $\mathbf{B}$ is said to be good, if the vectors $\vec{b_i}$ are short and nearly orthogonal. For any basis $\mathbf{B}$ it holds that

$$\prod_{i=1}^{n} ||b_i|| \geq \det(\Lambda) \qquad \text{(Hadamard's inequality)}$$

So, good bases come closer to equality.

Definition 2.34. The **dual lattice** of $\mathcal{L}$, denoted $\mathcal{L}^*$, is defined as

$$\mathcal{L}^* := \{\vec{x} \in \text{span}(\mathbb{B}) : \forall \vec{v} \in \mathcal{L}, \langle \vec{x}, \vec{v} \rangle \in \mathbb{Z}\},$$

where $\langle \cdot, \cdot \rangle$ denotes an inner product.

Example. The dual lattice of $(\mathbb{Z}^n)^*$ is $(\mathbb{Z}^n)$ itself. For $(2\mathbb{Z}^n)^* = \frac{1}{2}\mathbb{Z}^n$, this is because one possible lattice basis is $Id = [\vec{e_1}, ..., \vec{e_n}]$ then for any $\vec{x} \in \mathcal{L}^*$ it must hold that $\langle \vec{x}, 2\vec{e_i} \rangle \in \mathbb{Z}$. That is the reason why it is sometimes called the *reciprocal lattice.*

Theorem 2.35. For a full-rank basis $\mathbf{B}$ for the lattice $\mathcal{L}$, $(\mathbf{B}^{-1})^T$ is a basis for its dual lattice $\mathcal{L}^*$.

Proof. Let us denote the lattice generated by the inverse transpose of B as $\mathcal{L}' = \mathcal{L}((\mathbf{B}^{-1})^T)$.
For any $\vec{y} \in \mathcal{L}((\mathbf{B}^{-1})^T)$ it holds that $\vec{y} \in (\mathcal{L}(\mathbf{B}))^*$. Since, for any $\vec{v} \in \mathbb{Z}^n$,

$$((\mathbf{B}^{-1})^T\vec{y})^T\mathbf{B}\vec{v} = \vec{y}^T\mathbf{B}^{-1}\mathbf{B}\vec{v} = \langle \vec{y}, \vec{v} \rangle \in \mathbb{Z}.$$

This proves $\mathcal{L}' \subseteq \mathcal{L}^*$. On the other hand, for any $\vec{y} \in (\mathcal{L}(\mathcal{B}))^*$, set $\vec{s}^T = \vec{y}^T\mathbf{B}$ then per definition $\vec{s} \in \mathbb{Z}^n$. Observing that $\vec{y}^T = \vec{s}^T\mathbf{B}^{-1}$, $\vec{y}$ is an integer linear combination of the rows of $\mathbf{B}^{-1}$, this proves $\mathcal{L}' \supseteq \mathcal{L}^*$ □

Let $\mathcal{B}_m(\vec{0}, r) = \{\vec{x} \in \mathbb{R}^m : \|\vec{x}\| < r\}$ be the $m-$dimensional open ball of radius r centered in $\vec{0}$. Fundamental constants associated with any rank n lattice Λ are its **successive minima** $\lambda_1, ..., \lambda_n$:

Definition 2.36. The i^{th} minimum $\lambda_i(\Lambda)$ is the radius of the smallest sphere centered in the origin containing i linearly independent lattice vectors

$$\lambda_i(\Lambda) = \inf\{r : \dim\left[\text{span}\left(\Lambda \cap \mathcal{B}\left(\vec{0}, r\right)\right)\right] \geq i\}.$$

Note that successive minima can be defined with respect to any norm.
For $p \geq 1$, the l_p norm of a vector $\vec{x} \in \mathbb{R}^n$ is $\|\vec{x}\|_p = (\sum_{i=1}^n x_i^p)^{\frac{1}{p}}$. Special cases are

- the l_1-norm $\|\vec{x}\|_1 = \sum_{i=1}^n |x_i|$,
- the **euclidean norm** $l_2 = \|\vec{x}\|_2 = \sqrt{\langle x, x\rangle} = \sqrt{\sum_{i=1}^n x_i^2}$,
- the **max-norm** $l_\infty = \|\vec{x}\|_\infty = \lim_{p\to\infty} \|\vec{x}\|_p = \max_{i=1}^n |x_i|$.

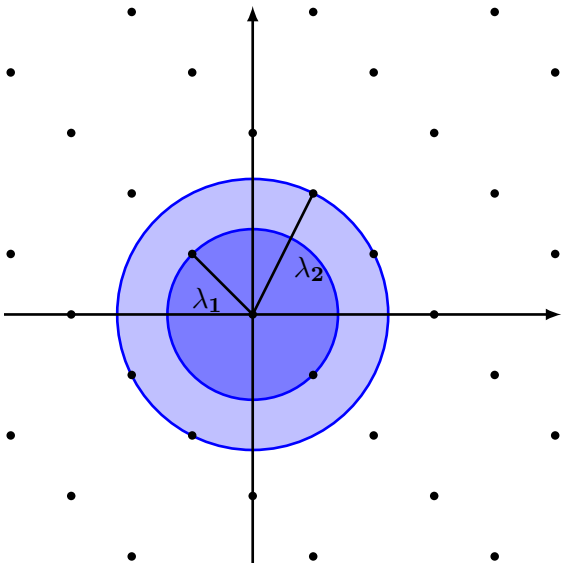

Figure 2.3.: The first two successive minima λ_1, λ_2

In his 1910 published book *Geometrie der Zahlen*[33] Hermann Minkowski provided the fundamental *convex body theorem*, from which an upper bound of the product of successive minima can be derived.

Theorem 2.37 (Convex body theorem)**.** For any lattice Λ of rank n and any convex set $S \subset \text{span}(\Lambda)$ symmetric about the origin, if $\text{vol}(S) > 2^n \det(\Lambda)$, then S contains a nonzero lattice point $\vec{v} \in S \cap \Lambda \setminus \{\mathbf{0}\}$.

This statement can be used to bound the length of the shortest nonzero vector of an n-ranked lattice Λ. Let S be the open ball centered in the origin with radius $\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$ in span(λ). The n-dimensional hypercube that fits into the ball with radius $r = \sqrt{n}\det(\Lambda)^{\frac{1}{n}}$ has edges of length $a = \frac{2 \cdot r}{\sqrt{n}} = 2\det(\Lambda)^{\frac{1}{n}}$. The volume of

the hypercube is $a^n = 2^n \det(\Lambda)$. Therefore the volume of S is strictly bigger than $2^n \det(\Lambda)$. By theorem 2.37 there exists a non-zero lattice vector $\vec{v}$ such that $\vec{v} \in S$, hence $\|\vec{v}\| < \sqrt{n} \det(\Lambda)^{\frac{1}{n}}$. This proves that for any rank n lattice Λ, the length of the shortest nonzero vector (in the l_2-norm) satisfies

$$\lambda_1 < \sqrt{n} \det(\Lambda)^{\frac{1}{n}}$$

However, the value of the λ_i, and the lattice vectors achieving them depend on the norm being used.
There always exist vectors achieving the successive minima:

Theorem 2.38. There are n linearly independent vectors $\vec{x}_1, ..., \vec{x}_n \in \Lambda$, such that $\|\vec{x}_i\| = \lambda_i$, for all $i = 1, ..., n$.
The proof of this theorem can be found in [32], page 9.

In particular, λ_1 is the length of the shortest nonzero lattice vector and equals the minimum distance between any two distinct lattice points

$$\lambda_1(\Lambda) = \min_{\vec{x} \neq \vec{y} \in \Lambda} \|\vec{x} - \vec{y}\| = \min_{\vec{x} \in \Lambda \setminus \{\mathbf{0}\}} \|\vec{x}\|.$$

Let $f(x) \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree n, e.g. $f(x) = x^n - 1$ where n is a power of 2. R denotes the polynomial ring $\mathbb{Z}[x]/(f(x))$ of integer polynomials modulo $f(x)$, e.g. with $f(x) = x^n - 1$, an element $g \in R$ is a polynomial of degree at most $n-1$. This polynomial is naturally associated to a coefficient vector in $\mathbb{Z}^n$, thus each element of R can be seen as a polynomial and a vector.
Let $I \trianglelefteq R$ be an ideal. Since I is additively closed, the coefficient vectors associated to elements of I form a lattice. I is called an **ideal lattice** to emphasize this objects dual nature as an *algebraic ideal* and a *lattice*.

Definition 2.39. Let $R = \mathbb{Z}[x]/(f(x))$ be the ring of integer polynomials modulo some monic polynomial $f(x)$ of degree n. Since R is isomorphic to $\mathbb{Z}^n$ as an additive group and ideals in R are by definition subgroups, they correspond to lattices. Lattices of this form are called **ideal lattices** with *respect to* f.

2.2. Complexity theory

Computational complexity theory focuses on classifying computational problems according to their computational hardness. Similar to the following theory of public key systems, the field of computational complexity theory has developed rapidly in the past three decades, due to the fact that all cryptographic systems rely on the intractability of an infeasible underlying computational problem. The central task of computational complexity theory is whether tasks can be performed efficiently

or not. The needed *tools* for this theory were provided by the computability theory, which emerged in the 1930s. This theory focuses on computational tasks and algorithms, and considers automated procedures that may solve such tasks. To unravel the complete theory of computational complexity, computability and problems would go beyond the scope of this thesis. An elaborate study on computational complexity theory can be found in [38]. In the following section a short introduction of the most necessary notions of complexity can be found.

2.2.1. Computational Problems

The purpose of complexity theory is to determine the amount of computational resources required to solve computational problems and to classify problems according to their difficulty. There are different options available on which resource should be focused on. The resource most often used is *time*. Another possible computational resource is memory (*space*). In the following, the resource focused on is *time*, since the complexity classes used for cryptographic purposes are almost always time dependent.
Before defining complexity classes, it is crucial to understand the fundamental objects of computational complexity theory:

A **computational problem** is a mathematical object representing a collection of questions that computers might want to solve. The input string for a computational problem is referred to as an *instance*. Respectively the output string is called the *solution*. A computational problem consists of an infinite amount of tuples which are composed of instances and the according solution.

There are two major fields which deal with computational problems. First, there is the field of *algorithm* research which is the study of methods for solving the problems efficiently. Second, there is the field of complexity theory which explains *why* a problem is believed to be unsolvable or intractable even if a great deal or infinite computational resources are available.

There are different types of computational problems of which the two most common will be further explained:

Search problem: A search problem consists of an infinite set of instances and a concise specification of valid solutions.

Example. *Factoring* a composite number is a search problem where the instance is a number n and the valid solution is a set of prime numbers $p_1, ..., p_n$ which are the prime factors to the number.

Decision problem: A decision problem consists of an infinite set of instances and a concise specification of YES-instances.

Example. *Primality testing* is a decision problem where the instance is a positive integer n and the problem is to determine if n is a prime number or not.

Another needed subject is the **model of computation**. Informally speaking it is a scientific model of computers which are used to determine the intractability of a computational problem. Each model differs in the number of computations which can be carried out by itself and the respective costs. In terms of cryptographic systems the used model of computation is the *Turning machine*. It is affiliated to the group of infinite automaton, which simple means infinite memory. Concerning the timeliness of the 1935 described machine, a very important thesis was formed in the first half of the 20^{th} century: The **Church-Turing Thesis** states that *any real-world computation can be translated into an equivalent computation involving a Turing machine*. There exists no proof for that thesis although every realistic model of computation discovered so far has been shown to be equivalent. A device which could answer questions beyond those that a Turing machine can answer would be called an **oracle**.

For the definition of the two complexity classes **P** and **NP** the differentiation between deterministic and nondeterministic Turing machines is needed.
Informally speaking a **nondeterministic Turing machine** (NTM) is a Turing machine which, no matter what state it is in, can take any action selecting from a set of specified actions. This is opposed to a **deterministic Turing machine** (DTM) which takes *one* predetermined action for each state it is in. So a NTM may take different actions at different times even if in the same situation.

Complexity Classes

The **time complexity** of a problem is the number of steps that it takes to solve an instance of the problem using the most efficient algorithm. In order to measure the time efficiency of a function the **Big-O Notation** is used. It is used to describe an *asymptotic upper bound* for the magnitude of a function in terms of another, usually simpler, function.

Definition 2.40 (Big-O Notation)**.** Suppose $f(x)$ and $g(x)$ are two functions defined on some subset M of the real numbers $\mathbb{R}$. Then
$f(x) = O(g(x))$ as $x \to \infty$ if and only if there exist real numbers x_0 and a positive real number k such that $|f(x)| \leq k \cdot |g(x)|$ for $x > x_0$.

Computational problems fall into sets of comparable complexity, called **complexity classes**. A complexity class is a set of functions that can be computed within a given time (formal resource). It is defined by the following 3 factors:

1. The **computational problem**. Oftentimes these are decisional problems but

they are also defined based on other computational problems like search problems or optimization problems, etc.

2. The **model of computation**. Mostly this is a Turing machine, but like above there are numerous complexity classes based on other models like Boolean circuits etc.

3. The **resource**. There are two most common types of resources, *time* and *space*. Throughout this thesis the used resource will be time.

The following table shows some complexity classes for *decision problems*:

Complexity class	**Model of Computation**	**Time constraint**
DTIME($f(n)$)	Deterministic Turing machine	$f(n)$
NTIME($f(n)$)	Non-deterministic Turing machine	$f(n)$
P = DTIME($n^{O(1)}$)	Deterministic Turing machine	n^k, polynomial time
NP = NTIME($n^{O(1)}$)	Non-deterministic Turing machine	n^k
EXP = DTIME($2^{n^{O(1)}}$)	Deterministic Turing machine	$2^{(n^k)}$, exponential time

Table 2.1.: Important complexity classes of decision problems in respect to the resource *Time*

The class **P** consists of all decision problems that can be *solved* on a deterministic machine in an amount of time that is polynomial in the size of the input.
The class **NP** on the other hand consists of all decision problems whose positive solutions can be *checked* in polynomial time, given the right information. Clearly, it holds that $\mathbf{P} \subseteq \mathbf{NP}$ but it stays an open question whether

$$\mathbf{P} = \mathbf{NP}$$

There are two other classes which should be introduced when talking about cryptographic security.
The complexity class **NP**-hard and **NP**−complete. But first it is necessary to define another complexity class function.

Definition 2.41. Let A and B be two decision problems. A **reduction** from A to B is a polynomial time computable function $f : \Sigma^* \to \Sigma^*$, where Σ^* is the set of all possible input strings, such that $x \in A$ if and only if $f(x) \in B$. If A reduces to B and B can be solved in polynomial time, then A can also be solved in polynomial time.
A decision problem A is **NP**-hard if any other **NP** problem B reduces to A. If A is also in **NP**, then A is **NP**-complete. Clearly, if a problem A is **NP**-hard, then A cannot be solved in polynomial time unless $\mathbf{P} = \mathbf{NP}$.

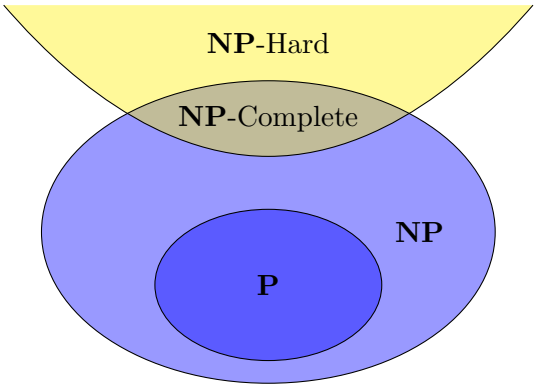

Figure 2.4.: The complexity classes: **P**, **NP**, **NP**-hard, **NP**-complete

2.2.2. The Integer Factorization problem

It is a fact that the security of many public-key cryptosystems relies on the apparent intractability of computational problems. This section presents the oldest and most famous of such problems, *the integer factorization problem*:
The integer factorization problem has been the subject of intense research, especially in the years since the invention of RSA in 1978. The problem was first described by Carl Friedrich Gauß in 1801 in [18] as: *"The problem of distinguishing prime numbers from composite numbers, and of resolving the latter into their prime factors, is known to be one of the most important and useful in arithmetic [...]"*. The problem itself can be described as [8]:

Definition 2.42. Let N be a composite integer. Find the integers u, v such that $N = u \cdot v$ and such that both $u, v > 1$. u and v are called *factors*.

Factoring a composite integer is believed to be a hard problem. The *general number field sieve* is the best known algorithm for factoring large n. For a b-bit number n the running time of the general number field sieve is about $O(e^{(\frac{64}{9}b)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}})$, i.e. sub exponential.
In 1994, Peter Shor discovered a *quantum algorithm* [40] which solves the factorization problem in polynomial time, namely $O(b^3)$.

When thinking about the complexity class the factorization problem is in, it is necessary to define the computational problem which is equivalent to the integer factorization problem stated above. The according decision problem version is:

Definition 2.43 (Computational Integer Factorization Problem, **FACT**)**.** Given an integer N and an integer M with $1 \leq M \leq N$, does N have a factor d with $1 < d < M$?

The exact class in which this problem falls into is not known yet, however it is known that it lies in **NP** because the YES and NO answers can be easily verified given the prime factors. Furthermore, it is suspected to be outside of the class **P**.
When mastering this problem, the *prime factorization* of a composite integer can be obtained recursively!

Remark. On December 12, 2009, a group of researchers from various countries factored a 768-bit, 232-digit number RSA-768 with the number field sieve [27]!

2.2.3. The Composite Residuosity Problem

The next problem that is introduced, focuses on n^{th} residues modulo a square number. It is the basis for the Paillier cryptosystem described in Chapter 6. It will be shown that the hardness of the computational problem of deciding n^{th}-residues is even stronger than the before defined integer factorization problem (see ch. 6, sec. refPaillier:intract, page 64).

Definition 2.44. A number z is said to be a $\mathbf{n^{th}}$ **residue modulo** $\mathbf{n^2}$, if there exists a number $y \in \mathbb{Z}^*_{n^2}$ such that

$$z = y^n \bmod n^2$$

The problem of deciding n^{th} residuosity, i.e., distinguishing n^{th}-residues from non n^{th}-residues will be denoted by $CR[n]$:

Definition 2.45 (Decisional Composite Residuosity problem, $\mathbf{CR[n]}$)**.** Given a composite n and an integer z, decide whether z is a n^{th}-residue modulo n^2, i.e., if there exists an integer y such that $z = y^n \bmod n^2$.

At the time of writing this thesis, there exists no polynomial time distinguisher for n^{th}-residues modulo n^2, i.e., $\mathbf{CR[n]}$ is intractable.

2.2.4. Lattice Problems

Minkowski's convex body theorem (Theorem 2.37) provides a simple but not constructive way to bound the length of λ_1 of the shortest nonzero vector in a lattice Λ, $\lambda_1 < \sqrt{n}\det(\Lambda)^{\frac{1}{n}}$. The convex body theorem is asymptotically tight in the worst case, i.e. there exist lattices such that $\lambda_1 < c\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$ for some constant c independent of n. In general, λ_1 can be much smaller than the convex body theorem bound.
The most classical and most studied problem on lattices is the problem of finding a lattice vector of length λ_1: the well known fundamental algorithmic *Shortest Vector Problem*.

Definition 2.46 (Shortest Vector Problem, **SVP**)**.** Given a basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, find a nonzero lattice vector $\mathbf{B}\vec{x}$ (with $\vec{x} \in \mathbb{Z}^n \setminus \{0\}$) such that $\|\mathbf{B}\vec{x}\| \leq \|\mathbf{B}\vec{y}\|$ for any other $\vec{y} \in \mathbb{Z}^n \setminus 0$.

Till the day of writing this thesis, there is no known polynomial time algorithm to solve *SVP*. In fact, it is not even known how to find nonzero lattice vectors of length within the *Minkowskis bound*.
In order to study the computational complexity of this problem we need another formulation of the SVP.

Definition 2.47 (Decisional Shortest Vector Problem, **DSVP**)**.** Given a rational $r > 0$, decide whether there is a (nonzero) lattice vector $\vec{x}$ such that $\|\vec{x}\| \leq r$.

Another related problem for which no polynomial time solution is known is the *Closest Vector Problem*.

Definition 2.48 (Closest Vector Problem, **CVP**)**.** Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ and a target vector $\vec{t} \in \mathbb{Z}^m$, find a lattice vector $\mathbf{B}\vec{x}$ closest to the target $\vec{t}$, i.e., find an integer vector $\vec{x} \in \mathbb{Z}^n$ such that $\left\|\mathbf{B}\vec{x} - \vec{t}\right\| \leq \left\|\mathbf{B}\vec{y} - \vec{t}\right\|$ for any other $\vec{y} \in \mathbb{Z}^n$.

And the associated computational problem:

Definition 2.49 (Decisional Closest Vector Problem, **DCVP**)**.** Given a rational $r > 0$, decide whether there is a (nonzero) lattice vector $\vec{x}$ such that $\left\|\vec{x} - \vec{t}\right\| \leq r$.

The hardness of solving **SVP** and **CVP** has led scientists to consider approximation versions of these problems. Approximation algorithms return solutions that are only guaranteed to be within some specified factor α from the optimum. Approximation versions for the **SVP** and **CVP** can be found in [32].
Regarding the complexity of the **SVP** Micciancio proved that SVP is **NP**-hard to solve even approximately, for any approximation factor up to $\sqrt{2}$ [31]. The decision problem associated to **CVP** is **NP**-complete, and therefore no algorithm can solve **CVP** in deterministic polynomial time, provided that $\mathbf{P} \neq \mathbf{NP}$.

The γ-*bounded distance decoding problem*(γ-BDDP) which is related to the **CVP** is more more important for our purpose. In fact, it is the same as the CVP with the promise that there is indeed a unique solution.

Definition 2.50 (γ-**Bounded Distance Decoding Problem**)**.** Given a basis $\mathbf{B}$ for a lattice $\mathcal{L}$ of dimension n and a vector $\vec{t} \in \mathbb{R}^n$ such that $\text{dist}(\mathcal{L}, \vec{t}) \cdot \gamma \leq \lambda_1(\mathcal{L})$, find the nonzero vector $\vec{v} \in \mathcal{L}$ closest to $\vec{t}$

$$||\vec{t} - \vec{v}|| \leq \gamma \cdot \text{dist}(\mathcal{L}, \vec{t}).$$

Liu et al. [29] showed that the γ-BDDP is **NP**-hard for any constant factor $\gamma > 1/\sqrt{2}$ in general lattices, they also showed that it can be solved in polynomial time for $\gamma = \mathcal{O}(\sqrt{(}\log(n)/n))$.

In general, the best known polynomial-time approximation algorithms are variants of the lattice reduction algorithm LLL [28], or two algorithms invented by L. Babai [2] (nearest plane, rounding off). These algorithms only work for exponential approximation factors. In fact, the following conjecture drawn from [39] holds

Conjecture 2.51. *Approximating these lattice problems to within a factor of* 2^k *takes time about* $2^{n/k}$.

2.3. Measure Theoretic Probability

This section will give a very short introduction to measure theory with special focus on the consequential notions for probability theory. It is kept rather short due to the fact that this thesis only requires the basic definitions to understand the usage of probability measures in Chapter 3. The definitions are taken from [25].

This section starts with some general notions and later show how these are used in the context of probability.

Definition 2.52. Let $\mathcal{F}$ be a collection of subsets of Ω, $\mathcal{F} \subset 2^{\Omega}$. $\mathcal{F}$ is called a **field** (*algebra*) if $\Omega \in \mathcal{F}$ and $\mathcal{F}$ is closed under complementation and finite union. That is,

1. It holds that $\Omega \in \mathcal{F}$
2. For every $A \in \mathcal{F}$ it holds that $A^c \in \mathcal{F}$
3. For every sequence $A_1, A_2, ..., A_n \in \mathcal{F}$ it holds that $\bigcup_{i=1}^{n} A_i \in \mathcal{F}$.

In addition, if the sequence in 3 can be enhanced by countable unions, that is if

3' For every sequence $A_1, ..., A_n, ... \in \mathcal{F}$ it holds that $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$,

then $\mathcal{F}$ is called a $\sigma -$ **algebra** .
If $\mathcal{F}$ is a $\sigma - algebra$ on Ω, then $(\Omega, \mathcal{F})$ is called a **measurable space** and the elements of $\mathcal{F}$ are called **measurable sets**.

Example. Consider the following examples:

i. The family of trivial sets $\mathcal{F} = \{\emptyset, \Omega\}$ is a $\sigma - algebra$ on Ω,
ii. The power set of a set Ω is a $\sigma - algebra$ on Ω, $\mathcal{F} = \{2^{\Omega}\}$,
iii. The family of sets $\mathcal{F} = \left\{A \in 2^{\Omega} \,|A \text{ or } A^c \text{ is countable}\right\}$ is a $\sigma - algebra$ on Ω

Definition 2.53. Let $(\Omega, \mathcal{F})$ be a measurable space. A **measure** on this space is a function $\mu : \mathcal{F} \to [0, \infty]$ with the properties

1. $\mu(\emptyset) = 0$,
2. if $A_i \in \mathcal{F}$ are disjoint then

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i).$$

The triple $(\Omega, \mathcal{F}, \mu)$ is called a **measure space**. If $\mu(\Omega) = 1$ it is called a **probability measure**, often denoted as P, and the according triple $(\Omega, \mathcal{F}, P)$ a **probability space**.

$$\mathrm{P} : \mathcal{F} \to [0, 1]$$

A measure τ is called the **counting measure** if it satisfies the following property: $\tau(A) = |A|$, the cardinality of A, hence the counting measure is only really very useful in finite measure spaces.
Ω is often called the *set of outcomes* and an element $\omega \in \Omega$ is called **outcome**. The elements of $\mathcal{F}$ are called **events**. So, by definition, an event is a measurable subset of the set of all outcomes and $\mathrm{P}(A)$ with $A \in \mathcal{F}$ is called the probability of A.

Example (1). The following experiment consists of tossing three coins. Each coin has individual outcomes 0 and 1. The set Ω can be written as
$\Omega = \{000, 001, 010, 100, 011, 101, 110, 111\}$, in this case we take $\mathcal{F} = 2^{\Omega}$ and a choice of P could be such that P assigns probability $\frac{1}{8}$ to all singletons, i.e., $(010) = \frac{1}{8}$.
In general, when having a finite set of outcomes $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ the usual choice of $\mathcal{F}$ is the power set of Ω, 2^{Ω} and a possible measure would be the counting measure $\mu(A) = |A|$. Furthermore, using the following probability measure $\mathrm{P}(A) = \mu(A)/|\Omega|$, the counting probability measure gives us the discrete uniform distribution.

Example (2). Let us consider an *infinite sequence* of coin tosses. In this case $\Omega = \{0, 1\}^{\mathbb{N}}$ and an element $\omega \in \Omega$ is an infinite sequence $(\omega_1, \omega_2, ...)$ with $\omega_n \in \{0, 1\}$. Let us say that one would like to have that sets like "*the* 4^{th} *outcome is* 0" are events. To achieve this, let us consider C to be the collection of all such sets, $C = \{\{\omega \in \Omega : \omega_n = s\}, n \in \mathbb{N}, s \in \{0, 1\}\}$. We take $\mathcal{F} = \sigma(C)$ and all sets $\{\omega \in \Omega : \omega_n = s\}$ are then events. One can show that there indeed exists a probability measure P on this $\mathcal{F}$ with the nice property that for instance the set $\{\omega \in \Omega : \omega_1 = \omega_2 = \omega_3 = 1\}$ has probability $\frac{1}{8}$ (in the previous example it would have been denoted by $\{111\}$).

3. Public-key Systems

This chapter presents the basic definitions for a public key cryptosystem. There are many formal definitions for public key cryptosystems which can be found in [22, 24, 30]. This chapter is a compendium of the most important definitions taken from the three books mentioned.

A different term for public-key cryptography is *asymmetric cryptography*, due to the asymmetric setting concerning the key information held by the participants. Namely one participant has a secret key, while the others have access to the public key that matches the secret key. This is in contrast to the symmetric setting, where there is only one key which must be handled with total secrecy by the two communicators. Whilst in the asymmetric setting there is only one secret key holder which makes it less vulnerable. Due to the more complex systems underlying the schemes, the computation of public key systems tend to be more time consuming. That is the reason why symmetric systems like AES are still in wide use.

The basic idea of a public-key encryption scheme was first proposed by Diffie and Hellman [12] in 1976 at Stanford University. Their idea was to enable secure message exchange between parties without ever having to meet in reality to agree on a common secret. They proposed the concept of a *trapdoor function* and outlined how it can be used to achieve such a secure cryptosystem. Shortly thereafter, *Rivest, Shamir and Adelman* proposed their method which enables secure message exchange. The scheme is called *RSA* after the names of its inventors (see Chapter 5).

Notation: When describing the assignment/outcome of an algorithm, the notation "$b \leftarrow \mathsf{A}(x)$" is used throughout this thesis - in this case the algorithm A outputs b on input x. An algorithm is denoted in sans serif form, e.g., $\mathsf{A}, \mathsf{B}, \mathsf{C}$....

A **polynomial-time algorithm** (PT) is an algorithm which runs in polynomial time, that is, if the number of steps required to complete the algorithm for a given input is $O(n^k)$, where n is the input size and k is a nonnegative integer. Any algorithm whose running time cannot be bound this way is called an *exponential-time algorithm.*
Roughly speaking, polynomial-time algorithms can be equated with good or *efficient algorithms*, while exponential-time algorithms are considered inefficient.

3.1. Modern Encryption

In modern cryptography the assumption that an adversary has infinite computation resources available is discarded, it is instead assumed that the adversary has bounded computation resources in some reasonable way. Therefore, the focus lies on hard computational problems rather than provable secure schemes, like *one-time-pads*. As a consequence, the term *infeasibility* of breaking a system is used instead of *impossibility* of breaking it.
In the majority of publications of the last decade it is assumed that the algorithms used are of probabilistic nature (see Section 3.2) and run in polynomial time (**PPT**). The running time of the encryption, decryption and the adversary algorithms are all measured as a function of the security parameter k.

As modern cryptography needs to combine *efficient algorithms* with *computational infeasibility* of decryption for the adversary, it requires that one has available **primitives** with certain special kinds of computational hardness properties. Of these, perhaps the most fundamental one is a *one-way function*.

Definition 3.1. A **negligible function** is a function $f : X \to Y$, if for every positive polynomial $p(\cdot) \in \mathbb{Z}[k]$ there exists an ε so that for all integers $n > \varepsilon$ it holds that $f(n) < \frac{1}{p(n)}$.
A function $f : \{0,1\}^n \to \{0,1\}^n$ is called **one-way function**, if the following two conditions hold:

1. (**Easy to compute:**) There exists a polynomial-time algorithm A computing f; that is, $\mathsf{A}(x) = f(x) = y$ for all x;

2. (**Hard to invert:**) For every polynomial-time algorithm B, there is a negligible function $\nu_{\mathsf{B}}(k)$ so that for sufficiently large k,

$$P\left[\mathsf{B}\left(f(x)\right) = x\right] \leq \nu_{\mathsf{B}}(k) \quad (\textit{negligible probability}), \tag{3.1}$$

   where in Equation (3.1) the probability is defined as in *Example (1)* on page 29. The set of outcomes Ω is the set of all possible n-long $0, 1$ stings. With an adequate σ-algebra, the event in question would be the set $\{\omega_j \in \Omega : \omega_j = x\}$ with $\omega_j = \mathsf{B}\left(f(x)\right)$ and $x \in \Omega$.

In Goldwasser and Bellare's words [22] the definition above *considers the success probability of an algorithm to be negligible if, as a function of the input length, the success probability is bounded by any polynomial fraction.*

Informally spoken, a function is one-way if it is *easy to compute* but *hard to invert*. In a public key setting such one-way functions are called *trapdoor functions*. The key holder possesses some trapdoor information, which makes it possible to invert

the function. Thus (1) and (2) of definition 3.1 holds true for everyone but the key holder, who can invert efficiently with the knowledge of the trapdoor information. Based on such primitives, it is possible to build secure encryption schemes.

However, it remains unknown if there exist any (true) *one-way functions*. No one has yet definitively proved the existence of such functions under reasonable definitions of "easy" and "computationally infeasible". The question of existence of such one-way functions leads deep into the field of complexity theory and in fact even further. If the existence of such functions would be proven, it would imply $\mathbf{P} \neq \mathbf{NP}$.
Of course, the same applies to *trapdoor one-way functions*. But there are a number of good candidates.

A List of Candidate One Way Functions

As mentioned above, it is crucial for a secure public key system to have functions which are easy to compute but hard to invert. *Easy* means that the function can be computed by a probabilistic polynomial time algorithm, denoted **PPT**. *Hard* means that any *PPT* attempting to invert the function will succeed with *negligible* probability. There are several candidates which seem to posses the needed properties. See [22, 24] for more than the following three candidates.

1. **Factoring**. The function $f : (x, y) \mapsto xy$ is conjectured to be a one way function. There are various algorithms for factoring numbers, but all of them run in at best expected time. The fastest known algorithm is the number field sieve.
2. **Discrete logarithm**. Let p be a prime. Then $\mathbb{Z}_p^*$ is cyclic, that means $\mathbb{Z}_p^* = \{g^i \bmod p | 1 \leq i \leq p-1\}$ for some generator $g \in \mathbb{Z}_p^*$. The function $f : (p, g, x) \mapsto (g^x \bmod p)$ is conjectured to be a one-way function. Computing $f(p, g, x)$ can be done in polynomial time using repeated squaring. The best known algorithm for computing the discrete logarithm is called the *index calculus algorithm* and runs in sub exponential time.
3. **RSA**. Let $n = pq$ be a product of primes. It is believed that such an n is hard to factor. The function $f(x) = x^e \bmod n$ where e is relatively prime to $\varphi(n)$. The trapdoor information is the knowledge of p, q which allows to invert f efficiently.

3.2. Probabilistic and Deterministic Encryption

A public-key encryption scheme is said to be *deterministic* if its encryption algorithm is deterministic. The notion "*deterministic encryption*" was introduced by Bellare et al. in [3]. Given a particular input, a deterministic algorithm **will always**

produce the same output and it will **always proceed in the same way**.

Deterministic encryption cannot achieve the classical notions of security of probabilistic encryption, but [3] formalized a semantic security style notion *PRIV* that captures the best possible privacy achievable when encryption is deterministic. Bellare et al. [4] defined seven notions of privacy for deterministic encryption. This thesis will not give an overview of those notions, instead it will focus on probabilistic encryption for following reasons.
Goldwasser and Bellare summarized three major problems when using a deterministic encryption algorithm in [22]:

1. *Special Message Spaces.* The fact that f is a deterministic function does not imply that inverting $f(m)$, when m is special, is hard. Suppose that the set of messages that one would like to send is drawn from a highly structured message space such as the English language, or more simply $M = \{0, 1\}$, it may be easy to invert $f(m)$. In fact, it is always easy to distinguish between $f(0)$ and $f(1)$.
2. *Partial Information.* The fact that f is a one-way or trapdoor function does not necessarily imply that $f(m)$ hides all information about m. Even a bit of leakage may be too much for some applications. Moreover, in fact, for any one-way function f, information such as "the parity of $f(m)$" about m is always easy to compute from $f(m)$.
3. *Relationship between Encrypted Messages.* Clearly, one may be sending messages which are related to each other in the course of a communication. It is thus desirable and sometimes essential that such dependencies remain secret. In the deterministic encryption model, it is trivial to see that sending the same message twice is always detectable.

Figure 3.1.: Schematic representation of a deterministic algorithm

For example, the original RSA scheme (see Chapter 5) is of deterministic nature.

Probabilistic Encryption

Probabilistic encryption algorithms were first mentioned in [23]. Their presented scheme is based on computational complexity theory and therefore the intractability of some problems in number theory, such as factoring or deciding quadratic residuosity with respect to composite moduli.

The basic idea is to endow an algorithm with the ability to generate random numbers[1]. As pointed out in the previous section, there are certain drawbacks of deterministic encryption, which all arise from the fact that a specific plaintext m will always be encoded in one specific ciphertext c_m. To avoid this, **probabilistic encryption schemes** utilize randomness within the encryption process itself, so that for one specific plaintext m there are many possible ciphertexts, $c_1, c_2, ..., c_r$. With the right secret key every possible ciphertext $c_1, c_2, ...c_r$ of a message will be decoded to the original message m!

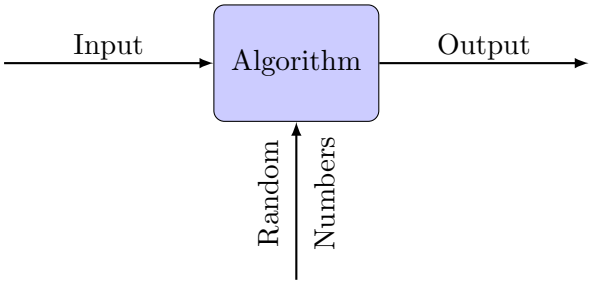

Figure 3.2.: Schematic representation of a probabilistic algorithm

It is however possible to convert a deterministic scheme into a randomized scheme by extending each plaintext with a randomly generated bit-string of a predefined length l, where l should be chosen sufficiently large. But note that the resulting randomized encryption scheme is generally not as secure as the probabilistic schemes.

3.3. Definition of a Public-Key Cryptosystem

For the definition and the setting of a public-key cryptosystem this thesis closely follows Katz and Bellares definition in [24]. Before stating the actual definition it is necessary to explain one very important variable, the **security parameter**.

The running time of the encryption, decryption, and the adversary algorithms are all measured as a function of a security parameter k which is a parameter that is fixed at the time the cryptosystem is setup. Thus, when it is stated that the adversary's algorithm runs in polynomial time, it means time bounded by some polynomial function in k. The security parameter k is usually expressed as a k-long string of bits 1.

An *asymmetric* cryptosystem is a method for secure communication between parties

[1] I don't want to pursue the issue where these random numbers actually come from. There a numerous authors who discuss that issue in more detail that I can make room here in this thesis

who have never met before. More precisely, a public key cryptosystem is composed of the following algorithms [24]:

Definition 3.2. A *public-key encryption scheme* $\mathcal{E}$ is a tuple, $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms

(1) The **key generation algorithm** (KeyGen) takes the security parameter k as input and outputs a pair of keys (pk, sk). I refer to the first of these as the **public key** and the second as the **private key** or **secret key**. I assume that pk and sk each have length at least k, and that k can be determined from pk, sk.

(2) The **encryption algorithm** (Enc) takes a public-key pk and a string m called the *message* from some underlying message space ($\mathbf{M}$) as input. It produces a ciphertext c from an underlying ciphertext space ($\mathbf{C}$), denoted as $c \leftarrow \mathsf{Enc}_{pk}(m)$ or simple $\mathsf{Enc}(m)$, if it is obvious which public key is in use.

(3) The **decryption algorithm** (Dec) takes a private-key sk and a ciphertext c as input, and produces an output message m. Without loss of generality we assume that Dec is deterministic, and write this as $m := \mathsf{Dec}_{sk}(c)$.

Remark. The definition above (Def. 3.2) states the encryption algorithm to be probabilistic. Some schemes, for the most part the older schemes, use encryption algorithms that are deterministic, i.e., RSA (see Ch. 5).

3.4. Notions of Security

When talking about cryptography, security is an inevitable subject. Menezes et al. state five models for evaluating security in [30]: Unconditional security, complexity-theoretic security, provable security, computational security and ad hoc security. Furthermore, they believe that the most practical security metrics are computational, provable and ad hoc security.

1. *Computational*: the amount of computational effort required,
2. *Provable*: the difficulty of defeating a cryptographic method can be shown to be essentially as difficult as solving a well-known and supposedly difficult (typically number-theoretic) problem,
3. *ad hoc*: convincing arguments which show that every successful attack requires a resource level (e.g., time and space) greater than the fixed resources of a perceived adversary.

Remark. Since this thesis focuses on the development of homomorphic cryptography the above mentioned security definitions will not be described in detail. The interested reader is referred to Menezes et al.: *Handbook of applied cryptography* [30].

The basic idea of computational security

Kerckhoffs' principle [26] consists of six design principles for *military* ciphers. One of which is very relevant to this thesis:

> *A [cipher] must be practically, if not mathematically, indecipherable.*

Katz and Lindell [24] interpret this principle as follows: "...,this principle of Kerckhoffs essentially says that it is not necessary to use a perfectly-secure encryption scheme, but it instead suffices to use a scheme that cannot be broken in "reasonable time" with any "reasonable probability of success",...".
The computational approach incorporates two relaxations of the notion of perfect security:

1. Security is only breached when adversary's algorithms run in a feasible amount of time.
2. Adversaries can potentially succeed with some *very small probability.*

Katz and Lindell state two common approaches to precisely define what is meant by the above. I will only write about the more important one, the **asymptotic approach**.
In this approach the running time of the adversary's algorithm, as well as its success probability, are considered to be functions of the security parameter k.
The notion of *efficient algorithms* are equated with probabilistic algorithms running in time *polynomial in k.* The notion of *small probability of success* is equated with success probabilities smaller than an inverse polynomial in k. This means for any constant c the adversary's success probability is smaller than k^{-c} for large enough values of k. A function that grows more slowly than any inverse polynomial is called *negligible.*
It is now possible to define asymptotic security.

Definition 3.3. A scheme is **secure** if every PPT adversary succeeds in breaking the scheme with only negligible probability.

Security of an encryption scheme, symmetric or asymmetric, reflects the inability of an adversary, given ciphertexts and any public information such as a public key or the security parameter, to get non-trivial information about the underlying plaintexts. Several desirable properties can be found throughout the various books concerning security of encryption schemes. The following paragraph will present three of the most common such properties.

1. The private key should not be recoverable from seeing the public key.
2. No useful information can be computed about messages from their encrypted form.

3. It should be hard to detect simple but useful facts about traffic of messages, such as when the same message is sent twice.

There are plenty of definitions proposed so far and most of them have been shown to be equivalent. For further information see [22, 23]. Under the above stated assumptions, consider a communication between Alice and Bob, schematically shown in Figure 3.3.

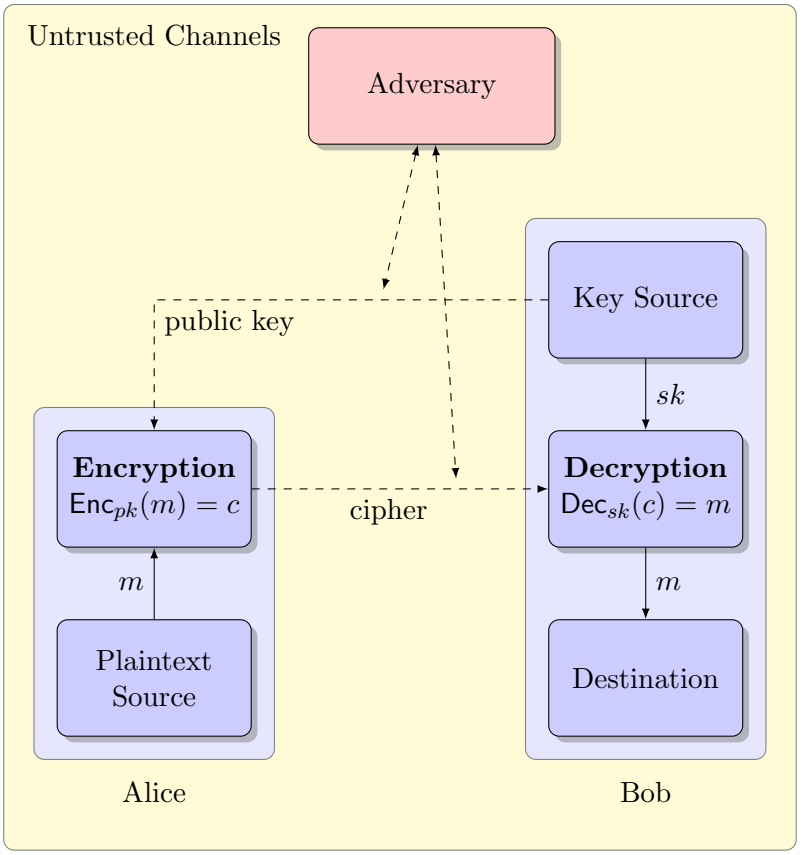

Figure 3.3.: Diagram of a communication using public-key techniques

4. Homomorphic Encryption

The security requirements for data and algorithms have become very strong in the last few years. Due to the vast growth of technology, a great variety of attacks on digital goods and technical devices are enabled. For storing and reading data securely there exist several possibilities such as secure data encryption. The problem becomes more complex when asking for the possibility to compute (publicly) with encrypted data or to modify functions in such a way that they are still executable while our privacy is ensured. That is where homomorphic cryptosystems can be used.

The notion and idea of *fully homomorphic schemes* was introduced by Rivest, Adleman and Dertouzos in [36] shortly after the invention of RSA [37]. They asked for an encryption function that permits encrypted data to be operated on without preliminary decryption of the operands, and they called those schemes *privacy homomorphisms*. Even in 1978 this was a highly important matter, it is even more important nowadays.
While the partially homomorphic properties of schemes like RSA, Paillier, ElGamal, etc. have been acknowledged ever since, it was not before 2009 when a young IBM researcher published the first working fully homomorphic cryptosystem, based on lattices.

4.1. Definition of a homomorphic encryption scheme

Chapter 3 focused on the formal definition (defn. 3.2) of a public key cryptosystem and the related definitions and notions. Now, in order to perform operations on ciphertexts, a *partially* or *fully* homomorphic system requires a fourth algorithm [24].

Definition 4.1. A public-key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is **homomorphic** if for all k and all (pk, sk) output from $\mathsf{KeyGen}(k)$, it is possible to define groups $\mathbf{M}, \mathbf{C}$ so that:

- The plaintext space $\mathbf{M}$, and all ciphertexts output by Enc_{pk} are elements of $\mathbf{C}$.
- For any $m_1, m_2 \in \mathbf{M}$ and $c_1, c_2 \in \mathbf{C}$ with $m_1 = \mathsf{Dec}_{sk}(c_1)$ and $m_2 = \mathsf{Dec}_{sk}(c_2)$ it holds that:
$$\mathsf{Dec}_{sk}(c_1 * c_2) = m_1 * m_2$$

where the group operations $*$ are carried out in $\mathbf{C}$ and $\mathbf{M}$, respectively.

In other words, a homomorphic cryptosystem is a PKS with the additional property that there exists an efficient algorithm (Eval) to compute an encryption of the sum or/and the product, of two messages given the public key and the encryptions of the messages, but not the messages themselves.
Moreover, a **fully homomorphic** scheme is able to output a ciphertext that encrypts $f(m_1, ..., m_t)$, where f is any desired function, which of course must be efficiently computable. No information about $m_1, ..., m_t$ or $f(m_1, ..., m_t)$, or any intermediate plaintext values should leak. The inputs, outputs and intermediate values are always encrypted and therefore useless for an adversary. Before we take a closer look on fully homomorphic encryption schemes, we will need another important notion from information theory.

Circuits

Informally speaking, **circuits** are *directed, acyclic graphs* where nodes are called **gates** and edges are called **wires**. Depending on the nature of the circuit the input values are integers, boolean values etc. and the corresponding gates are set operations and arithmetic operations or logic gates (AND, OR, NOR, NAND, ...). In order to evaluate a function f, express f as a circuit and topologically arrange its gates into levels which will be executed sequentially.

Example. Assume the function f outputs the expression $A \cdot B + B \cdot C \cdot (B + C)$ on input (A, B, C). Then the following circuit represents the function f, with the logic gates AND and OR.

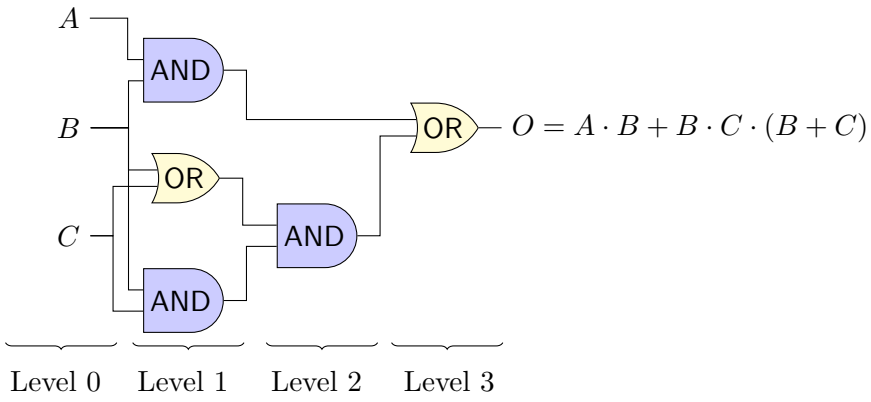

Figure 4.1.: Example for circuit representation

Two important complexity measures for circuits are **size** and **depth**.

Definition 4.2. The **size** of a circuit C is the number of its non-input gates.
The depth of a circuit C is the length of its longest path, from an input gate to the output gate, of its underlying directed graph.

This yields to another definition of fully homomorphic encryption [20]:

Definition 4.3 (**fully homomorphic encryption**)**.** A public key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is **fully homomorphic** if there exists an additional efficient algorithm Eval that, for a valid public key pk, a permitted circuit C and a set of ciphertexts $\Psi = \{c_1, ..., c_t\}$ where $c_i \leftarrow \mathsf{Enc}_{pk}(m_i)$, outputs

$$\mathbf{c} \leftarrow \mathsf{Eval}_{pk}(C, \Psi)$$

under pk.

There is another way to construct fully homomorphic encryption schemes which will be discussed later in this thesis. To understand how this transformation works, we need the following definitions and corollaries.

Definition 4.4 (**correct**)**.** A homomorphic encryption scheme $\mathcal{E}$ is said to be **correct** for a family $\mathcal{C}_{\mathcal{E}}$ of circuits if for any pair (sk, pk) output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$ any circuit $C \in \mathcal{C}_{\mathcal{E}}$, any plaintext $m_1, ..., m_t$, and any ciphertexts $\Psi = \langle c_1, ..., c_t \rangle$ with $c_i \leftarrow \mathsf{Enc}_{pk}(m_i)$, it is the case that:

$$\text{if } c \leftarrow \mathsf{Eval}_{\mathcal{E}}(pk, C, \Psi), \text{ then } \mathsf{Dec}_{\mathcal{E}}(sk, c) \rightarrow C(m_1, ..., m_t)$$

except with negligible probability over the random coins in $\mathsf{Eval}_{\mathcal{E}}$.

Gentry, the author of the fully homomorphic encryption scheme described in Chapter 7, additionally demands for a fully homomorphic encryption scheme to be **compact**, to exclude trivial schemes.

Definition 4.5 (**compact**)**.** A homomorphic encryption scheme $\mathcal{E}$ is **compact**, if there is a polynomial f so that, for every value of the security parameter λ, $\mathcal{E}$'s decryption algorithm can be expressed as a *circuit* $D_{\mathcal{E}}$ of size at most $f(\lambda)$.
A homomorphic encryption scheme $\mathcal{E}$ **compactly evaluates** circuits in $\mathcal{C}_{\mathcal{E}}$ if $\mathcal{E}$ is *compact* and also *correct* for circuits in $\mathcal{C}_{\mathcal{E}}$.

Corollary 4.6. *A homomorphic encryption scheme $\mathcal{E}$ is **fully homomorphic** if it compactly evaluates all circuits.*

This demand is considered to be almost too strong for practical purposes, hence he uses a certain relaxation to include leveled schemes, which only evaluate circuits of depth up to some d, and whose public key length may be poly(d).

Definition 4.7 (**leveled fully homomorphic**)**.** A family of homomorphic encryption schemes $\{\mathcal{E}^{(d)} : d \in \mathbb{Z}^+\}$ is said **leveled fully homomorphic** if, for all $d \in \mathbb{Z}^+$, they all use the same decryption circuit, $\mathcal{E}^{(d)}$ compactly evaluates all circuits of depth at most d (that use some specified set of gates), and the computational complexity of $\mathcal{E}^{(d)}$'s algorithms is polynomial in λ, d, and (in the case of $\mathsf{Eval}_{\mathcal{E}}$) the size of the circuit C.

An encryption scheme which supports both addition and multiplication (a fully homomorphic scheme) thereby preserves the ring structure of the plaintext space, and is therefore far more powerful. Using such a scheme makes it possible to let an untrusted party do the computations without ever decrypting it and therefore preserving the privacy of the data.
As an application they suggested private data banks. A subject which would exceed this work, therefore I refer the interested reader to the paper itself [36].

A widely esteemed application of homomorphic encryption schemes is **cloud computing**. Presently, the need for cloud computing is increasing fast, as the data we are processing and computing on is getting bigger and bigger every day, with the effect that a single person's computation power does not suffice anymore. Hence, it is favorable to use someone else's power without losing the privacy we seek.

Say, Alice wants to store a sensitive file $m \in \{0,1\}^n$ on Bob's server. So she sends Bob $\mathsf{Enc}(m_1), ..., \mathsf{Enc}(m_n)$. Assume that the file is a database (a list of people with specific informations about them) and Alice wants to find out how many of them are 22 years old. Instead of retrieving the data from Bob, decrypting it and searching for the wanted information, she will ask Bob to do the computations, without him knowing what or who he is computing on. The answer from Bob comes in form of a ciphertext which only she can decrypt with her secret key.

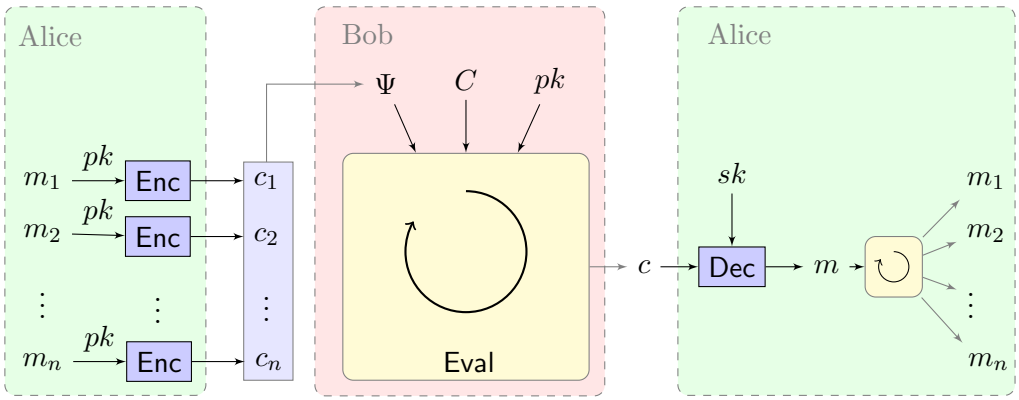

Figure 4.2.: Diagram of a homomorphic encryption scheme

4.2. Example of an additively homomorphic scheme

The Paillier encryption scheme, which will be described in detail in Chapter 6, is an example of a homomorphic encryption scheme over an additive group taking $\mathbf{M} = (\mathbb{Z}_n, +)$ and $\mathbf{C} = (\mathbb{Z}_{n^2}^*, \cdot)$.

One may ask why this is a useful feature. One practical application of Paillier's additive homomorphic property can be found in cryptographic voting schemes like the following. How to obtain the encryptions and decryptions will be shown in section 6.7 with the exact same numbers. The focus of this example shall be the practical usefulness of the additive homomorphic property of Paillier's scheme.

Let us assume the election will take place with

$$N_v = 9 \ldots \text{ Number of voters}$$
$$N_r = 5 \ldots \text{ Number of candidates}$$

Vote messages for candidates will be denoted with the use of an appropriate base b:

$$\begin{aligned} &1^{st} \text{ candidate:} && b^0 \\ &2^{nd} \text{ candidate:} && b^1 \\ &\vdots \\ &N_r^{th} \text{ candidate:} && b^{N_r-1} \end{aligned}$$

Now let us imagine the following outcome of a small election, where the V_i's, $i = 1, ..., 9$, are the voters and the R_i's, $i = 1, ..., 5$, are the candidates. For each voter it is possible to vote 0 to 2 candidates with only one vote for each candidate.

	R_1 10^0	R_2 10^1	R_3 10^2	R_4 10^3	R_5 10^4	message to be encrypted
V_1		✓				$m = 10^1 = 10$
V_2			✓		✓	$m = 10^2 + 10^4 = 10100$
V_3						$m = 0$
V_4				✓		$m = 10^3 = 1000$
V_5	✓			✓		$m = 10^0 + 10^3 = 1001$
V_6		✓		✓		$m = 10^1 + 10^3 = 1010$
V_7			✓	✓		$m = 10^2 + 10^3 = 1100$
V_8		✓		✓		$m = 10^1 + 10^3 = 1010$
V_9	✓					$m = 10^0 = 1$
Total	2	3	2	5	1	

Table 4.1.: A simple voting example with $N_v = 9$ voters and $N_r = 5$ candidates. The base used is $b = 10$

The maximum possible number representing a voter's single vote m_{max} can exceed to

$$m_{max} = \sum_{i=1}^{N_r} b^{i-1}.$$

Hence, the possible maximum tally of all votes can exceed to:

$$T_{max} = N_v \cdot m_{max}.$$

Concerning the public key generation, n must satisfy the following condition to be able to encrypt the maximum tally:

$$n \geq T_{max} + 1, \text{ where } n = pq$$

Assume the voting authority chooses $p = 293$ and $q = 433$, then $n = 126869$ is the public key visible to everyone. Paillier's encryption function uses a random number $r \in \mathbb{Z}_n^*$ which can be chosen randomly by the voter itself. The detailed encryption process will be described in Chapter 6.

	vote only known by the voter	**random number** $\in \mathbb{Z}_n^*$	ciphertext c_i
V_1	10	369	4946672768
V_2	10100	6498	3355936313
V_3	0	1354	4336831183
V_4	1000	6957	7446214290
V_5	1001	265	3283050915
V_6	1010	34	4821154392
V_7	1100	659	4760329430
V_8	1010	1312	5720727730
V_9	1	444	11626554097
Tally	$T = \prod_{i=1}^{N_v} c_i \bmod n^2$		10631213431

Table 4.2.: The encrypted votes of table 4.1. Note that although voter V_6 and V_8 have voted the same, the encryption of their votes are different.

As seen in the table above the tally is the product of all ciphertexts, $T = \prod_{i=1}^{N_v} c_i \bmod n^2$. This computation should be done by an external observer. This guarantees that the authority will not decipher each encrypted vote to see for which candidate he or she voted. Since the external observer only handles encrypted data without the knowledge of the secret key, the computation can be done publicly.
Now the tally will be sent back to the authority, the only one who can decipher the

encrypted version of all votes.

$$\mathsf{Dec}(T) = 15232$$
$$\text{which is indeed } = \sum_{i=1}^{N_v} m_i \bmod n$$

Now the election authority wants to know who has won the election. This will be done by converting the tally, which is now in decimal form, to a number with the base chosen in the beginning ($b = 10$)[1]

$$15232 = 1 \cdot 10^4 + 5 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0$$

Candidate R_4 is the winner.

In a nutshell, the additive homomorphic property which made the election above perfectly secret is

$$\underbrace{\prod_{i=1}^{N_v} c_i \bmod n^2}_{\in(\mathbb{Z}^*_{n^2},\cdot)} = \underbrace{\sum_{i=1}^{N_v} m_i \bmod n}_{\in(\mathbb{Z}_n,\cdot)}.$$

[1] As we used the base 10 there is no conversion needed. Nevertheless it is stated in case a different base is used.

5. RSA - A Multiplicatively Homomorphic Scheme

In 1978, Rivest, Shamir, and Adleman published their public-key cryptosystem, which only uses elementary ideas from number theory, in their paper *"A Method for Obtaining Digital signatures and Public-Key Cryptosystems"* [37]. It was one of the first homomorphic cryptosystem. The RSA cryptosystem is the most widely used public-key cryptosystem. It may be used to provide both secrecy and digital signatures and its security is based on the intractability of the integer factorization problem (see Section 2.2.2).
The following section describes the RSA encryption scheme and its security. The original RSA paper [37] can be seen as the foundation to this chapter. Further occurring statements will be cited directly.

5.1. The Definition of the RSA Cryptosystem

In this section the scheme itself is described. The **key generation** procedure outputs a public and a secret key. The public-key consists of two integers e and n, where n is a composite of two large primes p, q chosen by the secret key holder. The second integer e has to be chosen such that the greatest common divisor of e and $\varphi(pq)$ is $\gcd(e, \varphi(pq)) = 1$, namely e is invertible mod $\varphi(n)$. Here $\varphi(n)$ denotes the *Euler totient function* (see Definition 2.11). Hence with the knowledge of p and q the Euler function can be easily computed as $\varphi(n) = (p-1)(q-1)$. The secret key is the tuple (d, n), where d is determined such that d is the inverse of e. This can be done by using the *extended euclidean algorithm*. A detailed description of this specific algorithm can be found in [41, chap 3].

The **encryption** algorithm takes as input a message m from the plaintext space $\mathbb{Z}_n$ and computes the according ciphertext $c = m^e \bmod n$. This integer $c \in \mathbb{Z}_n$ can not be traced back to the original message without the knowledge of p and q, which will be proved later in this section.

Decryption takes as input the ciphertext c and the secret key (d, n) and computes $m = c^d \bmod n$. Since d is the inverse of e in $\mathbb{Z}_n$ this is indeed the original message.

The three steps (key generation, encryption and decryption) can be found in table 5.1

Key Generation: $\mathsf{KeyGen}(p, q)$	
Input: $p, q \in \mathbb{P}$	
Compute	$n = p \cdot q$
	$\varphi(n) = (p-1)(q-1)$
Choose e such that	$\gcd(e, \varphi(n)) = 1$
Determine d such that	$e \cdot d \equiv 1 \bmod \varphi(n)$
Output: (pk, sk) public key: $pk = (e, n)$ secret key: $sk = (d)$	

Encryption: $\mathsf{Enc}(m, pk)$	
Input: $m \in \mathbb{Z}_n$	
Compute	$c = m^e \bmod n$
Output: $c \in \mathbb{Z}_n$	

Decryption: $\mathsf{Dec}(c, sk)$	
Input: $c \in \mathbb{Z}_n$	
Compute	$m = c^d \bmod n$
Output: $m \in \mathbb{Z}_n$	

Figure 5.1.: The RSA Cryptosystem

In order to see that the scheme above deciphers correctly it is necessary to prove that decryption really outputs the original message m.

Proof. Since $e \cdot d \equiv 1 \bmod \varphi(pq)$, there exists an integer k such that

$$e \cdot d = 1 + k \cdot \varphi(pq).$$

The greatest common divisor of m and p is either $\gcd(m, p) = 1$ or $\gcd(m, p) = p$ itself. In the former case ($\gcd(m, p) = 1$) Fermats Theorem (theorem 2.17) yields

$$m^{p-1} \equiv 1 \bmod p$$

Raising both sides to the power $k \cdot (q-1)$ and then multiplying both sides with m

results

$$m^{\overbrace{1+k(p-1)(q-1)}^{e\cdot d}} \equiv m \bmod p \tag{5.1}$$

Hence we have $m^{e\cdot d} \equiv m \bmod p$.
If $\gcd(m, p) = p$, then congruence (5.1) is valid since each side is congruent to 0 modulo p.
Therefore it holds for all messages $m \in \mathbb{Z}_n$

$$m^{e\cdot d} \equiv m \bmod p$$

The same applies to q.
Finally, since p and q are distinct primes and n is the product of p and q, it follows that

$$m^{e\cdot d} \equiv m \bmod n$$
$$c^d = (m^e)^d \equiv m \bmod n$$

which completes the proof. □

5.2. Multiplicative Homomorphic Property

As said before the RSA scheme has a multiplicative homomorphic property. This means it is possible to perform multiplications with the encryptions of messages without losing or tampering with their underlying information. This is possible since the operation "*multiplication*" in the ciphertext space $(\mathbb{Z}_n, \cdot)$ can be compared with the operation "*multiplication*" in the plaintext space $(\mathbb{Z}_n, \cdot)$.

Given $c_i = \mathsf{Enc}(m_i) = m_i^e \bmod n$ then following properties hold

c_1	$=$	$m_1^e \bmod n$	
c_2	$=$	$m_2^e \bmod n$	
$c_1 \cdot c_2$	$=$	$m_1^e \cdot m_2^e \bmod n$	$= (m_1 \cdot m_2)^e \bmod n$

Table 5.1.: The homomorphic property of the RSA Cryptosystem

Example. Assume the following RSA keys

$pk = (5437, 189781)$
$sk = (49269)$

and messages $m_1 = 56947$ and $m_2 = 64413$. Now encrypting the messages using the

public key yields

$$\begin{aligned} c_1 &= 56947^{5437} \bmod 189781 = 96068 \\ c_2 &= 64413^{5437} \bmod 189781 = 149380 \\ c_1 \cdot c_2 &= 96068 \cdot 149380 \bmod 189781 = \boxed{157744}. \end{aligned}$$

This is the same as

$$\begin{aligned} m_1 \cdot m_2 &= 56947 \cdot 64413 \bmod 189781 = 39943 \\ \mathsf{Enc}(m_1 \cdot m_2) &= 39943^{5437} \bmod 189781 = \boxed{157744}. \end{aligned}$$

5.3. Security of RSA

The security of the RSA scheme is strongly related to the well studied *Integer Factorization Problem* (see Section 2.2.2). In general for every public key cryptosystem there are different approaches on "breaking" a scheme's security. Breaking it can for example mean

- Deciphering the ciphertext without knowing sk.
- Computing the secret key from the public key only.
- Partially deciphering a ciphertext.

The first approach would be the task of taking e^{th} roots modulo n. This computational task is also called the **RSA Problem**.

Definition 5.1 (RSA Problem). Given n, an integer $e > 0$ that is relatively prime to $\varphi(n)$, and an element $y \in \mathbb{Z}_n^*$. Compute $y^{1/e} \bmod n$, i.e., find x such that $x^e = y \bmod n$.

At the time of writing the most promising approach is to factor n and compute the inverse of e, namely d. This approach asks one to solve an instance of the *integer factorization problem*, which is known to be computationally hard. In the list above, this approach reflects the second item: computing sk from pk.

In 1994 the American mathematician Peter Shor developed a quantum algorithm for integer factorization [40] simply called **Shor's algorithm**, which runs in polynomial time. Until quantum computers are possible, the fastest known classical algorithm for solving an instance of the integer factorization problem is the *general number field sieve*. However, for now it is a safe bet to rely on the security of RSA as long as

- n is chosen sufficiently large (i.e., $>$ 768-bit).
- e is sufficiently large.

- m is chosen such that $m^e > n$.

The requirements above are only a short list of means to avoid attacks on an RSA communication. Imagine the following scenario:
Alice sends the message $m = 5$ to 3 different receivers and uses $e = 3$ as the exponent used to encrypt. The adversary intercepts the three different ciphertexts

$$c_i = 5^3 \bmod n_i$$

Using the Chinese Remainder Theorem to solve

$$z = c_i \bmod n_i \text{ in } \mathbb{Z}_{n_1 \cdot n_2 \cdot n_3}$$

gives the adversary the original message by computing

$$m = \sqrt[3]{z}.$$

5.4. A Worked Example

For better understanding an example is given below. Note that the numbers used are rather small. In reality these numbers are much bigger, i.e. $n \geq 768$ bit, 232 digits. Assume Alice wants to send the message $m_A = 1275$ to Bob. The two of them have provided their public keys somewhere publicly and stored their private keys at home (safely).

Alice
$pk_A = (907, 186101)$
$sk_A = (2851)$

Bob
$pk_B = (5437, 189781)$
$sk_B = (49269)$

Figure 5.2.: The RSA keys of Alice and Bob

Only Alice resp. Bob can compute their secret keys, due to the assumption that factoring n is hard. But in this example we will calculate the secret key step by step.
Alice chooses two primes prior to publicizing her private key.

$$p = 149 \text{ and } q = 1249, \text{ hence } n = p \cdot q = 186101.$$

Then she computes $\varphi(n)$ which is easy since she knows the prime factors of n.

$$\varphi(pq) = (149 - 1) \cdot (1249 - 1) = 184704.$$

Following that she chooses an e such that

$$\gcd(e, 184704) = 1.$$

We clearly see that $e_A = 907$ fulfills this property. To get the secret key d she has to solve

$$907 \cdot d = 1 \bmod 184704.$$

She can quickly do this using the **extended Euclidean algorithm**:

$184704 = 203 \cdot 907 + 583$	$\Leftrightarrow$	$583 = \underline{1} \cdot 184704 - \underline{203} \cdot 907$
$907 = 1 \cdot 583 + 324$	$\Leftrightarrow$	$324 = \underline{1} \cdot 907 - \underline{1} \cdot 583$
$583 = 1 \cdot 324 + 259$	$\Leftrightarrow$	$259 = \underline{1} \cdot 583 - \underline{1} \cdot 324$
$324 = 1 \cdot 259 + 65$	$\Leftrightarrow$	$65 = \underline{1} \cdot 324 - \underline{1} \cdot 259$
$259 = 3 \cdot 65 + 64$	$\Leftrightarrow$	$64 = \underline{1} \cdot 259 - \underline{3} \cdot 65$
$65 = 1 \cdot 64 + 1$	$\Leftrightarrow$	$1 = \underline{1} \cdot 65 - \underline{1} \cdot 64 = \underline{2851} \cdot 907 - \underline{14} \cdot 184704$
$64 = 64 \cdot 1 + 0$		

Table 5.2.: Extended Euclidean algorithm

and therefore $sk_A = (2851, 186101)$.

Bob does the same and computes his secret key, $sk_B = (49269, 189781)$. Now Alice is able to send Bob her secret message $m_A = 1275$, additionally she *signs* her message with her secret key. She does this by taking her secret key and using it on her private message m_A

$$c_A = 1275^{2851} \bmod 186101 = 127296$$

followed by encoding c_A with Bob's public key $(49269, 189781)$

$$c_{AB} = 127296^{5437} \bmod 189781 = 182522.$$

This is the ciphertext she can send via unsecured channels to Bob.

Bob himself uses his secret key (49269) to decode Alice's message

$$m_{AB} = 182522^{49269} \bmod 189781 = 127296$$

and further deciphers it with Alice's public key $(907, 186101)$

$$m_A = 127296^{907} \bmod 186101 = 1275.$$

Bob has obtained Alice's secret message and will also know that it was Alice who encoded the message, since she is the only one who knows how to transform 1275 into 127296.

5.4.1. Encoding a real message with RSA

In the previous example the prime numbers used are rather small so it is easily possible to recalculate the computations.
What would it look like if Alice wanted to send Bob a real message not just any integer?

First of all a standard character-encoding scheme is necessary. There are numerous such schemes of which one of the most common ones is the *8-bit extended ASCII table* or *ISO 8859* [1]. It consists of 256 different characters which cover most languages that use the Latin alphabet. So if we want to convert a text message into a computable string we have to convert each character into the according string specified by the encoding standard used. There are different ways to combine these strings into one distinguishable string dependent on the numeral system used. For example the integer $a = 196$ with the use of either one of the following systems is represented as

- $a = 196$ with the **decimal system**, using integers $0, ..., 9$ and $b = 10$,
- $a = 11000100$ with the use of the **binary system**, which uses the integers $0, 1$ and therefore $b = 2$,
- $a = C4$ with the **hexadecimal system**, using the symbols $0, ...9, A, ..., F$ with a base $b = 16$.

Here we want to use the decimal system.

The message which shall be encoded is:

Adi Shamir

The first step is to encode this message into the according decimal number in the *ISO8859 standard* in the decimal system.

character:	A	d	i	$\sqcup$	S	h	a	m	i	r
decimal ISO representation:	65	100	105	173	83	104	97	109	105	114
digit position:	9	8	7	6	5	4	3	2	1	0

It is necessary to combine the used decimal representations to yield an integer that can be used in the already known RSA encryption scheme. As the ISO8859 standard has *256* distinguished characters, the basis for the integer representation is $b = 256$:

$$m = \sum_{i=0}^{n} ISOrep(a_i) * 256^{lp(a_i)}$$

where a_i is the ISO8859 character, $ISOrep(a_i)$ its corresponding ISO representation and $lp(a_i)$ the digit position number. This method is also called *the positional*

notation. Applied to this example

$$\begin{aligned}\mathbf{m} =& 65 \cdot 256^9 + 100 \cdot 256^8 + 105 \cdot 256^7 + 173 \cdot 256^6 + 83 \cdot 256^5 + 104 \cdot 256^4 + \\ & 97 \cdot 256^3 + 109 \cdot 256^2 + 105 \cdot 256^1 + 114 \cdot 256^0 = \mathbf{308806070940178907490674}.\end{aligned}$$

In order to usefully encrypt the message, n must exceed m. To achieve this the two prime factors p and q will be chosen rather big.

$$\begin{aligned}& p = 464327924040839 \text{ and } q = 545626836709961 \Rightarrow \\ & n = 253349776390506035635362097279 \\ & \quad \text{choose encryption exponent } e \text{ relatively prime to } \varphi(pq) \\ & e = 23^1 \text{ and } d = 231319361052200240838809925047\end{aligned}$$

Now to encrypt the message compute

$$\begin{aligned}c &= m^e \bmod n \\ &= 308806070940178907490674^{23} \bmod 253349776390506035635362097279.\end{aligned}$$

The integers in this example are much bigger than in the prior example but still smaller than in reality. Either way exponentiating is a very expensive operation. Fortunately there are methods for faster computation with less time consuming ways. For example **binary exponentiation** or **square-and-multiply**.

The general idea behind this method is the following observation

$$x^n = \begin{cases} x \cdot \left(x^{\frac{n-1}{2}}\right)^2 & \text{if } n \text{ is odd} \\ 1, & \text{if } n = 0 \\ \left(x^{\frac{n}{2}}\right)^2 & \text{if } n \text{ is even} \end{cases}$$

With this observation it is possible to create a recursive algorithm using only squaring and multiplying. In this example the exponent is $e = 23$. Using this idea, x^{23} can be broken down to

$$\begin{aligned}x^{23} =& x \cdot x^{22} = x \cdot \left(x^{11}\right)^2 = x \cdot \left(x \cdot x^{10}\right)^2 = x \cdot \left(x \cdot \left(x^5\right)^2\right)^2 \\ =& x \cdot \left(x \cdot \left(x \cdot x^4\right)^2\right)^2 = x \cdot \left(x \cdot \left(x \cdot \left(x^2\right)^2\right)^2\right)^2 \end{aligned} \tag{5.2}$$

In equation (5.2) we need to multiply 3 times and square 4 times in contrast to naively multiplying the base with itself 23 times. In words, the calculation sequence is: *square, square, multiply, square, multiply, square and finally multiply.* This yields the following computation

[1] In reality it is wise to choose a bigger e, since there are various attacks on small encryption exponents.

$\mathbf{c = m^{23} \bmod n}$

Step 1 **Square**
$$m_1 = m^2 \bmod n = 116817648318621128495369899557$$

Step 2 **Square**
$$m_2 = m_1^2 \bmod n = 139125216491532649817800323 72$$

Step 3 **Multiply**
$$m_3 = m_2 \cdot m \bmod n = 125300342823104533089446554622$$

Step 4 **Square**
$$m_4 = m_3^2 \bmod n = 134664074173794183149523451861$$

Step 5 **Multiply**
$$m_5 = m_4 \cdot m \bmod n = 901120570622472613219196361 05$$

Step 6 **Square**
$$m_6 = m_5^2 \bmod n = 247331899466514929976873542756$$

Step 7 **Multiply** $c = m_6 \cdot m \bmod n$

$\mathbf{c = 163270259402702795275875214006}$

Figure 5.3.: Square and Multiply

This ciphertext can be sent to Bob using unsecured channels. To decipher he simply exponentiates the ciphertext with the secret exponent $d = 231319361052200240838809925047$

$$c^{231319361052200240838809925047} \bmod 25334977639050603563536209727 9 = m$$

With the knowledge of the used encryption standard he is able to convert

$$m = 308806070940178907490674 = \textbf{Adi Shamir}.$$

6. Paillier - An Additively Homomorphic Scheme

Pascal Paillier introduced his cryptosystem in the 1999 published paper *"Public-Key Cryptosystems Based on Composite Degree Residuosity Classes"* [35]. The proposed technique is based on *composite residuosity classes*, whose computation is believed to be computationally difficult. It is a probabilistic asymmetric algorithm for public key cryptography and inherits additive homomorphic properties.

6.1. The Definition of Paillier's Cryptosystem

This section introduces the scheme itself. The **key generation** algorithm takes as input two large primes p and q. It then computes the composite $n = p \cdot q$ and chooses an integer $g \in \mathbb{Z}_{n^2}^*$ (i.e., g in invertible modulo n^2) such that n and $L(g^\lambda \bmod n^2)$ are coprime, where L denotes the function

$$\begin{aligned} L : \mathbb{Z}_{n^2}^* &\to \mathbb{Z}_n \\ u &\mapsto \frac{u-1}{n} \end{aligned}$$

and λ denotes the Carmichael function $\lambda(p \cdot q) = \operatorname{lcm}(p-1, q-1)$. The public key is the tuple (n, g) and the secret key are the two prime factors (p, q).

The **encryption** procedure takes as input a message $m \in \mathbb{Z}_n$ and randomly chooses an integer r in $\mathbb{Z}_n^*$, this random number is used to fulfill the probabilistic algorithm's property, that one plaintext can have many ciphertexts (see Section 3.2).It is later shown that this random variable does not impede the correct decryption, but has the effect of changing the corresponding ciphertext.
The output is a ciphertext in $\mathbb{Z}_{n^2}$, which has the following form

$$c = g^m \cdot r^n \bmod n^2$$

Decryption takes as input the ciphertext c and the secret key p and q. The message is retrieved by

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$$

Note that due to the requirement that n and $L(g^\lambda \bmod n^2)$ are coprime it is possible to find the inverse of $L(g^\lambda \bmod n^2) \bmod n$.

Key Generation: $\mathsf{KeyGen}(p, q)$	
Input: $p, q \in \mathbb{P}$	
Compute Choose $g \in \mathbb{Z}^*_{n^2}$ such that	$n = pq$ $\gcd(L(g^\lambda \bmod n^2), n) = 1$ with $L(u) = \frac{u-1}{n}$
Output: (pk, sk) public key: $pk = (n, g)$ secret key: $sk = (p, q)$	

Encryption: $\mathsf{Enc}(m, pk)$	
Input: $m \in \mathbb{Z}_n$	
Choose Compute	$r \in \mathbb{Z}^*_n$ $c = g^m \cdot r^n \bmod n^2$
Output: $c \in \mathbb{Z}_{n^2}$	

Decryption: $\mathsf{Dec}(c, sk)$	
Input: $c \in \mathbb{Z}_{n^2}$	
Compute	$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$
Output: $m \in \mathbb{Z}_n$	

Figure 6.1.: Probabilistic encryption scheme based on composite residuosity

6.2. Additive Homomorphic Property

The Paillier scheme is known to be additivly homomorphic (see Chapter 4). What might seem confusing at first is the fact that the two group operations are different, namely the *product* of two ciphertexts will decrypt to the sum of their plaintexts. In comparison to that, the product of two RSA ciphertexts decrypt to the product of their plaintexts. Hence the Paillier scheme is additivly homomorphic and RSA multiplicatively.

The given ciphertexts c_i are valid encryptions of plaintexts m_i, $c_i = \mathsf{Enc}(m_i) =$

$g^{m_i} r_i^n \bmod n^2$. The following properties hold

c_1	$=$	$g^{m_1} x_1^n \bmod n^2$	
c_2	$=$	$g^{m_2} x_2^n \bmod n^2$	
$c_1 \cdot c_2$	$=$	$g^{m_1} x_1^n \cdot g^{m_2} x_2^n \bmod n^2$	$= g^{m_1+m_2}(x_1 x_2)^n \bmod n^2$

Table 6.1.: The homomorphic property of the Paillier Cryptosystem

This means that the encryption of the addition of two plaintexts m_1 and m_2 is exactly the multiplication of the associated ciphertexts c_1 and c_2. This property is applied to ,i.e., voting schemes (see Section 4).

Example. For the example we use the following Paillier public key

$$(n, g) = (2501, 92)$$

and say we want to encrypt the two messages $m_1 = 34$ and $m_2 = 16$. The according ciphertexts are

$$\begin{aligned} c_1 &= 92^{34} \cdot \underbrace{5}_{r_1}{}^{2501} \bmod n^2 = 1129735 \\ c_2 &= 92^{16} \cdot \underbrace{7}_{r_2}{}^{2501} \bmod n^2 = 5140305 \\ c_1 \cdot c_2 &= 2010769 \end{aligned}$$

On the other hand, adding the two ciphertexts and encrypting them, with the random number being $r_{1+2} = 35$ yields

$$\begin{aligned} m_1 + m_2 &= 50 \\ c_{1+2} &= 92^{50} \cdot 35^{2501} \bmod n^2 = 2010769 \end{aligned}$$

In the calculations above the random number r_{1+2} is exactly chosen to be the product of r_1 and r_2. In reality the entity which handles the computations, has of course no knowledge of the random number picked by the encrypter. But remember that the random number is not important anymore when we decrypt the ciphertexts, so it is actually possible to use another random number and get the correct sum of the original messages. However, in this example the exact value for the random number is used, so it is possible to compare the ciphertexts themselves.

6.3. The $\mathbf{n^{th}}$ Residue

As the scheme's security relies on the intractability of distinguishing n^{th} residues modulo n^2 from non n^{th} residues, a detailed study of these numbers is more than useful. In the following chapter, n denotes a composition of two primes p and q.

Definition 6.1. A number $z \in \mathbb{Z}_{n^2}^*$ is said to be a **n^{th} residue modulo n^2** if there exists a number $y \in \mathbb{Z}_{n^2}^*$ so that

$$z = y^n \bmod n^2$$

y is called an n^{th}-root.

The **set of n^{th} residues**, denoted $\mathcal{R}_n$, is a multiplicative subgroup of $\mathbb{Z}_{n^2}^*$, since n divides the order of $\mathbb{Z}_{n^2}^*$, i.e. $|\mathbb{Z}_{n^2}^*| = \varphi(n^2) = n \cdot \varphi(n)$. Moreover the cardinality of $\mathcal{R}_n$ is $|\mathcal{R}_n| = \varphi(n)$.

There are several interesting properties within the structure of $\mathbb{Z}_{n^2}^*$ and $\mathcal{R}_n$, which will be used later when proving correctness of the scheme. The first one concerns the amount of different n^{th}-roots of an n^{th}-residue.

Theorem 6.2. Each n^{th} residue $z \in \mathbb{Z}_{n^2}^*$ has exactly n distinct n^{th} roots in $\mathbb{Z}_{n^2}^*$

Proof. In a finite cyclic group G it holds that the equation $y^n = z$ has $\gcd(n, |G|)$ solutions [9]. This fact can be applied to the cyclic groups $\mathbb{Z}_{p^2}^*$ and $\mathbb{Z}_{q^2}^*$. The order of $\mathbb{Z}_{p^2}^*$ is $|\mathbb{Z}_{p^2}^*| = p(p-1)$ respectively the order of $\mathbb{Z}_{q^2}^*$ is $|\mathbb{Z}_{q^2}^*| = q(q-1)$. Then the following equation

$$z = y^n \bmod p^2 \tag{6.1}$$

has $\gcd(n, p(p-1)) = p$ distinct solutions, respectively equation 6.1 with q instead of p has $\gcd(n, q(q-1)) = q$ different solutions. Using the Chinese remainder theorem (thm. 2.18, p. 9) gives us $pq = n$ different solutions modulo n^2. □

These roots have a special form

Lemma 6.3. For any $x \in \mathbb{Z}_n$,

$$(1+n)^x = 1 + xn \bmod n^2$$

These are the roots of unity in $(\mathbb{Z}_{n^2}^*, \cdot)$.

Proof. This will be proved by induction over x. For $x = 0$ it is obviously true. Inductive step $x \to x + 1$:

$$\begin{aligned}(1+n)^x &= (1 + x \cdot n) \bmod n^2 \\ (1+n)^{x+1} &= (1+n)^x(1+n) \bmod n^2 \\ &= (1 + x \cdot n)(1+n) = (1 + (x+1) \cdot n) \bmod n^2\end{aligned}$$

These are indeed roots of unity since raising them to the power n yields

$$\begin{aligned}(1+n)^{x \cdot n} &= 1 + (xn)n \bmod n^2 \\ &= 1 \bmod n^2\end{aligned}$$

Also, $(1+n)^x \in \mathbb{Z}_{n^2}^*$ since $\gcd(1+n, n^2) = \gcd(1+n, n) = 1$.

□

Note that there is exactly one root of unity which is smaller than n, namely 1, which can easily be verified with Lemma (6.3).

Example. Lets choose $n = 33$ then $n^2 = 1089$. For example 269 is an n^{th} residue modulo 1089, because $14^{33} \equiv 269 \bmod 1225$. Hence 14 is an n^{th} root of 269 in $\mathbb{Z}_{1089}$. More n^{th} roots of 269 are $47, 80, 113, ...$
For the following table remember $y \in \mathbb{Z}_{n^2}^*$ and there is exactly one root strictly smaller than n.

y	$y^n \bmod n^2$	y	$y^n \bmod n^2$	y	$y^n \bmod n^2$	y	$y^n \bmod n^2$
1	1	8	215	17	161	26	251
2	602	10	604	19	820	28	766
4	856	13	118	20	971	29	233
5	323	14	269	23	485	31	487
7	838	16	928	25	874	32	1088

Table 6.2.: The n^{th} residues of $n = 33$

6.4. Paillier's Encryption Function

In this section we will take a closer look at the encryption function Enc_g. Let g be some element of $\mathbb{Z}_{n^2}^*$ and Enc_g is the integer-valued function defined by

$$\begin{aligned} \mathsf{Enc}_g : \mathbb{Z}_n \times \mathbb{Z}_n^* &\longrightarrow \mathbb{Z}_{n^2}^* \\ \mathsf{Enc}_g(m, r) &= g^m \cdot r^n \bmod n^2 \end{aligned} \tag{6.2}$$

Depending on g, Enc_g may have some interesting properties.

Lemma 6.4. If the order of $g \bmod n^2$ is a nonzero multiple of n, then Enc_g is bijective.

Proof. Since the two sets $\mathbb{Z}_n \times \mathbb{Z}_n^*$ and $\mathbb{Z}_{n^2}^*$ have the same cardinality,

$$|\mathbb{Z}_n \times \mathbb{Z}_n^*| = n \cdot \varphi(n) = \varphi(n^2) = \left|\mathbb{Z}_{n^2}^*\right|$$

it is only necessary to prove *injectivity*.
Assume that $m_1, m_2 \in \mathbb{Z}_n$ and $r_1, r_2 \in \mathbb{Z}_n^*$ with

$$g^{m_1} r_1^n = g^{m_2} r_2 \bmod n^2. \tag{6.3}$$

Multiplying (6.3) with g^{-m_2} and r_1^{-n} and raising it to the power $\lambda(n)$ yields

$$(g^{m_1-m_2})^{\lambda(n)} = \left(\frac{r_2}{r_1}\right)^{n\cdot\lambda(n)} \bmod n^2$$

Due to equation (2.4) it holds that $(r_2/r_1)^{n\lambda(n)} = 1 \bmod n^2$ and therefore

$$g^{(m_1-m_2)\lambda(n)} = 1 \bmod n^2 \tag{6.4}$$

On basis of the prerequisite that the order of g is a multiple of n, we have

$$\begin{aligned} &\text{ord}(g)|(m_1-m_2)\lambda(n) \\ &\Rightarrow n|(m_1-m_2)\lambda(n) \\ &\Rightarrow n|(m_1-m_2), \end{aligned}$$

since $\gcd(\lambda(n), n) = 1$. It follows that $g^{\lambda(n)}$ has order n and therefore is a root of unity. Consequently it can be written as $g^{\lambda(n)} = (1+zn)$ for some $z \in \mathbb{Z}_n, z \neq 0$. Therefore equation (6.4) can be written as

$$(1+zn)^{m_1} = (1+zn)^{m_2} \bmod n^2$$

This implies that $m_1 = m_2 \bmod n$. Equation (6.3) simplifies to

$$\begin{aligned} r_1^n &= r_2^n \bmod n^2 \\ \left(\frac{r_1}{r_2}\right)^n &= 1 \bmod n^2 \end{aligned}$$

which leads to the unique solution $r_1/r_2 = 1$ in $\mathbb{Z}_n^*$, hence $r_1 = r_2 \bmod n$. This completes the proof. □

This theorem demonstrates how to choose g such that we can exactly retrace a ciphertext to its original plaintext. How to construct such a g is demonstrated in the following section.

6.5. The n^{th} Residue Class

How hard is it to compute the inverse of Paillier's encryption function (see Equation6.2)?
To get a useful scheme we need the order of g be a nonzero multiple of n (see Lemma 6.4). Therefore the following set is constructed:

Let $\mathcal{B}_\alpha \subset \mathbb{Z}_{n^2}^*$ be the set of elements of order $n\alpha$

$$\mathcal{B}_\alpha := \{g \in \mathbb{Z}_{n^2}^* \,|\, \text{ord}(g) = n\cdot\alpha,\, \alpha \in \{1,...,\lambda\}\}$$

and by $\mathcal{B}$ their disjoint union for $\alpha = 1, ..., \lambda$.

Lemma 6.5. The order of $g \in \mathbb{Z}_{n^2}^*$ is a multiple of n, i.e. $g \in \mathcal{B}$, if and only if $\gcd(L(g^{\lambda(n)} \bmod n^2), n) = 1$

Proof. Due to equation (2.4) it holds that

$$g^{\lambda(n)} = 1 \bmod n \text{ for all } g \in \mathbb{Z}_{n^2}^*.$$

Therefore there exists a $k \in \mathbb{Z}_n$ such that[1]

$$g^{\lambda(n)} = 1 + kn \bmod n^2.$$

This equation is raised to the power $a \in \mathbb{N}$, it follows that

$$g^{a\lambda(n)} \equiv (1 + kn)^a \equiv 1 + a \cdot kn \bmod n^2 \tag{6.5}$$

If $\gcd(k, n) = b > 1$, then there is an $a < n$ for which the equivalence in Equation (6.5) is 1. Hence $a\lambda(n) = \text{ord}(g)$ and since a is strictly smaller than n, it follows that $g \notin \mathcal{B}$.

Conversely, $\gcd(L(g^{\lambda(n)} \bmod n^2), n) = 1$. To get the order of g, it is necessary to check when the equivalence in equation (6.5) becomes 1. This is the case only if a is chosen such that $a \cdot k$ is a multiple of n, therefore since $\gcd(k, n) = 1$, a has to be a multiple of n. Hence, the order of g is a multiple of n. □

Definition 6.6. Assume that $g \in \mathcal{B}$. For $c \in \mathbb{Z}_{n^2}^*$, we call the **$\mathbf{n^{th}}$ residuosity class** of c with respect to g the unique integer $m \in \mathbb{Z}_n$ for which there exists $r \in \mathbb{Z}_n^*$ such that

$$\mathsf{Enc}_g(m, r) = c = g^m r^n \bmod n^2.$$

The class of c is denoted $[\![c]\!]_g = m$

There are some essential properties of the class $[\![c]\!]_g$.

Lemma 6.7. The residuosity class $[\![c]\!]_g = 0$ if and only if c is an n^{th} residue modulo n^2

Proof. If $[\![c]\!]_g = 0$, then there exists a $r \in \mathbb{Z}_{n^2}^*$, such that $c = g^0 r^n \bmod n^2$. Hence c is an n^{th} residue modulo n^2.
Conversely, if c is an n^{th} residue modulo n^2, it has the form $c = r^n \bmod n^2$. So for every $g \in \mathcal{B}$ the following condition holds

$$c = r^n \bmod n^2 = g^0 r^n \bmod n^2$$

Hence $[\![c]\!]_g = 0$ since Enc_g is injective. □

[1] The function L on input $g^{\lambda(n)} \bmod n^2$ provides this k.

Lemma 6.8. The function $c \mapsto [\![c]\!]_g$ is a homomorphism from $(\mathbb{Z}_{n^2}^*, \cdot)$ to $(\mathbb{Z}_n, +)$ for any $g \in \mathcal{B}$. That means for all $c_i \in \mathbb{Z}_{n^2}^*$ following condition holds

$$[\![c_1 \cdot c_2]\!]_g = [\![c_1]\!]_g + [\![c_2]\!]_g \bmod n$$

Proof. Assume that for every m_i there exists a $r_i \in \mathbb{Z}_{n^2}^*$ with

$$c_i = g^{m_i} r_i^n \bmod n^2 \text{ for } i \in \{1, 2\}$$

then for $c = c_1 \cdot c_2$ there exists a $r = r_1 \cdot r_2$ with

$$c_1 \cdot c_2 \equiv g^{m_1} r_1^n \cdot g^{m_2} r_2^n \equiv g^{m_1+m_2} (r_1 \cdot r_2)^n \bmod n.$$

which completes the proof □

To be a useful cryptographic scheme the encryption function has to be easily computable but hard to invert. Obviously the first contraint is fulfilled, but how hard is it to compute $[\![c]\!]_g$?

6.6. The Intractability of the Scheme

The security of Paillier's scheme relies on two computational problems. First there is the *Composite Residuosity Class Problem* which is described as

Definition 6.9. (CLASS$[n, g]$). Given $c \in \mathbb{Z}_{n^2}^*$ and g, compute $[\![c]\!]_g$.

The second problem is the *Composite Residuosity Problem*, informally it asks to decide n^{th} residuosity, i.e., distinguishing n^{th} residues from non n^{th} residues. It will be denoted as CR$[n]$.

Definition 6.10. (CR$[n]$). Given $c \in \mathbb{Z}_{n^2}^*$ decide whether c is an n^{th} residue or a random element in $\mathbb{Z}_{n^2}^*$.

Note that like the well studied problems of deciding quadratic or higher degree residuosity [34], deciding n^{th} residuosity is believed to be computationally hard. Therefore, we assume

Conjecture 6.11. *(Decisional Composite Residuosity Assumption (DCRA)). There exists no polynomial time distinguisher for n^{th} residues modulo n^2, i.e., CR$[n]$ is intractable.*

Before we can analyze the intractability of CLASS$[n]$, further propositions about this problem are needed.

6.6.1. CLASS$[n, g]$

First it will be shown that the complexity of CLASS$[n, g]$ is actually independent of g, therefore CLASS$[n, g]$ becomes CLASS$[n]$. Additionally that CLASS$[n]$ is equally hard for all $c \in \mathbb{Z}^*_{n^2}$, i.e. random self reducible over c.

Lemma 6.12. Class$[n, g]$ is *random self reducible* over $c \in \mathbb{Z}^*_{n^2}$

Proof. Any $c \in \mathbb{Z}^*_{n^2}$ can be transformed into a random instance $\tilde{c} \in \mathbb{Z}^*_{n^2}$ of CLASS$[n]$ simply by randomly choosing $a \in \mathbb{Z}_n$, $a > 0$ and $b \in \mathbb{Z}^*_n$

$$\tilde{c} = c \cdot g^a b^n \bmod n^2$$

Assume we can solve CLASS$[n, g]$ for $\tilde{c}$, i.e. $[\![\tilde{c}]\!]_g = \tilde{m}$, then there exists a $\tilde{x} \in \mathbb{Z}^*_n$ such that

$$\begin{aligned} \tilde{c} &= g^{\tilde{m}} \tilde{x}^n \mod n^2 \\ &= c \cdot g^a b^n \bmod n^2 \end{aligned}$$

which yields

$$c = g^{\tilde{m}-a} \left(\frac{\tilde{x}}{b}\right)^n \mod n^2$$

and therefore

$$[\![c]\!]_g = \tilde{m} - a = [\![\tilde{c}]\!]_g - a \mod n.$$

which completes the proof. □

So it is equally hard to compute the n^{th} residuosity class for each ciphertext, even independently of g:

Lemma 6.13. *Class*$[n, g]$ is random self reducible over $g \in \mathcal{B}$

Proof. The lemma basically means that

$$\forall g_i \in \mathcal{B} \text{ it holds that } Class[n, g_1] \equiv Class[n, g_2].$$

Now assume that for any $c \in \mathbb{Z}^*_n$ and $g_1, g_2 \in \mathcal{B}$ there exist $r_1, r_2 \in \mathbb{Z}^*_n$ such that

$$\begin{aligned} c &= g_1^{m_1} r_1^n \bmod n^2 \quad \text{with } m_1 := [\![c]\!]_{g_1} \\ c &= g_2^{m_2} r_2^n \bmod n^2 \quad \text{with } m_2 := [\![c]\!]_{g_2} \end{aligned} \tag{6.6}$$

Say that for $[\![g_2]\!]_{g_1}$ there exists a $x \in \mathbb{Z}^*_n$ such that

$$g_2 = g_1^{m_3} x^n \bmod n^2 \quad \text{with } m_3 := [\![g_2]\!]_{g_1} \tag{6.7}$$

Now we combine (6.6) and (6.7):

$$\begin{aligned} c &= (g_1^{m_3} x^n)^{m_2} r_2^n \bmod n^2 \\ &= g_1^{m_2 \cdot m_3} (x^{m_2} \cdot r_2)^n \bmod n^2 \\ \Rightarrow g_1^{m_1} r_1^n &= g_1^{m_2 \cdot m_3} (x^{m_2} \cdot r_2)^n \bmod n^2 \\ \Rightarrow m_1 &= m_2 \cdot m_3 \end{aligned}$$

Therefore, we have

$$\begin{aligned} [\![c]\!]_{g_1} &= [\![c]\!]_{g_2} [\![g_2]\!]_{g_1} \bmod n \\ \text{resp. } [\![c]\!]_{g_2} &= [\![c]\!]_{g_1} [\![g_2]\!]_{g_1}^{-1} \bmod n \end{aligned} \tag{6.8}$$

Now assume that we are able to compute $[\![c]\!]_{g_1}$ then from lemma (6.13) we know we can substitute c with any other element in $\mathbb{Z}_{n^2}^*$, this yields

$$[\![g_1]\!]_{g_2} = [\![g_1]\!]_{g_1} [\![g_2]\!]_{g_1}^{-1} \bmod n$$

since for each $g \in \mathbb{Z}_{n^2}^*$ it holds that $[\![g]\!]_g = 1$, we get:

$$[\![g_1]\!]_{g_2} = [\![g_2]\!]_{g_1}^{-1} \bmod n \tag{6.9}$$

thus $[\![g_1]\!]_{g_2}$ is invertible modulo n and we can compute

$$[\![c]\!]_{g_2} = [\![c]\!]_{g_1} [\![g_2]\!]_{g_1}^{-1} \bmod n$$

which completes the proof. □

The lemmas in the previous sections essentially mean that the complexity of CLASS$[n, g]$ is independent of g and c. This enables us to analyze it as a computational problem which purely relies on n.

6.6.2. The Computational Hierarchy of Paillier's Encryption Scheme

The first connection we will establish is between the *Composite Residuosity Class Problem* and a standard number-theoretic problem, namely *Factoring*. For this purpose the next Lemma is helpful.

Lemma 6.14. For any $c \in \mathbb{Z}_{n^2}^*$, $L(c^\lambda \bmod n^2) = \lambda [\![c]\!]_{1+n} \bmod n$.

Proof. Since $(1+n) \in \mathcal{B}$, there exists a unique pair $(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$ such that $c = (1+n)^a b^n \bmod n^2$. By definition, $a = [\![c]\!]_{1+n}$ and therefore

$$c^\lambda = (1+n)^{a\lambda} b^{n\lambda} = (1+n)^{a\lambda} = 1 + a\lambda n \bmod n^2$$

which yields

$$L(c^\lambda \bmod n^2) = L(1 + a\lambda n \bmod n^2) = \lambda a \bmod n = \lambda [\![c]\!]_{1+n} \bmod n.$$

□

As said before it is believed that solving an instance of CLASS[n] is computationally infeasible, which is shown in the following theorem.

Theorem 6.15. The problem of computing the n^{th} residuosity class is as hard as factoring a composite number n.

$$\text{CLASS}[n] \Leftarrow \text{Fact}[n]$$

Proof. Due to equation (6.9) the $[\![g]\!]_{1+n} = [\![1+n]\!]_g^{-1} \bmod n$ is invertible, and as a consequence of Lemma (6.14), $L(g^\lambda \bmod n^2)$ is invertible modulo n. Now, factoring n leads to the knowledge of $\lambda = \text{lcm}(p-1, q-1)$. Due to equation (6.8) it holds that for any $g \in \mathcal{B}$ and $c \in \mathbb{Z}_{n^2}^*$,

$$\frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} = \frac{\lambda[\![c]\!]_{1+n}}{\lambda[\![g]\!]_{1+n}} = \frac{[\![c]\!]_{1+n}}{[\![g]\!]_{1+n}} = [\![c]\!]_g \bmod n \tag{6.10}$$

That means that $[\![c]\!]_g \bmod n$ is easily computable with the knowledge of p, q.
Note that this theorem also proves correctness of decryption. □

To connect the *Composite Residuosity Problem* with the *Composite Residuosity Class Problem* we will first show the equivalence of the decisional version of CLASS[n] and CR[n]

Theorem 6.16. Let D-CLASS[n] be the decisional problem linked to CLASS[n], i.e.,

$$\text{Given } c \in \mathbb{Z}_{n^2}^*,\ g \in \mathcal{B} \text{ and } m \in \mathbb{Z}_n, \text{ decide whether } m = [\![c]\!]_g \text{ or not.}$$

Then

$$\text{CR}[n] \equiv \text{D-CLASS}[n].$$

Proof. Assume there is an oracle solving CR[n], then we could submit $c \cdot g^{-m} \bmod n^2$ to the oracle. Due to Lemma (6.7), in case of n^{th} residuosity, it holds

$$[\![c \cdot g^{-m}]\!]_g = 0$$

which implies

$$\begin{aligned} c \cdot g^{-m} &= g^{[\![c]\!]_g} r^n \cdot g^{-m} \bmod n^2 \\ &= g^{[\![c]\!]_g - m} r^n \bmod n^2 \end{aligned}$$

hence $[\![c \cdot g^{-m}]\!]_g = 0$ only if $[\![c]\!]_g = m$.
Conversely, there is an oracle solving D-CLASS[n]. Randomly choose $g \in \mathcal{B}$ and submit the triple $(c, g, 0)$ to the oracle. Due to Lemma (6.7), if the answer is YES so is the answer to CR[n]. □

Above we see the computational hierarchy, where D-CLASS$[n] \Leftarrow$ CLASS$[n]$ comes from the general fact that it is easier to verify a solution than to compute it.

$$\text{CR}[n] \equiv \text{ D-CLASS}[n] \Leftarrow \text{ CLASS}[n] \Leftarrow \text{ FACT}[n]$$

This leads to the second intractability hypothesis

Conjecture 6.17. *(Computational Composite Residuosity Assumption (CCRA)). There exists no probabilistic polynomial time algorithm solving the Composite Residuosity Class Problem, i.e., Class$[n]$ is intractable.*

6.7. A Worked Example

For better understanding and to show the probabilistic property of the scheme see the following example. Like in the example of the RSA scheme (see Section 5.4, page 51) the numbers used are much smaller than in reality.

The following example uses two messages from the voting example in Section 4.2. The two voters V_6 and V_8 will be named Alice and Bob. Alice's and Bob's votes are the same, they want either R_2 or R_4 to win the proposed election. As the votes need to be converted to numbers each candidate gets a different number. According to the example on page 43, $R_2 = 10^1$ and $R_4 = 10^3$. The message which shall be encrypted is the vote $m = 1010$. This message is the same for both Alice and Bob, since they voted the same candidates.
Now, the authority picks two primes $p = 293$ and $q = 433$ secretly and calculates the public key $n = p \cdot q = 126869$ and $n^2 = 16095743161$. Only the authority can compute the Carmichael function, since it knows the prime factors of n, $\lambda(n) = \text{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1,q-1)} = 31536$. For the completion of the public key the authority chooses an integer $g \in \mathbb{Z}^*_{16095743161}$ with the property that $L(g^\lambda \bmod n^2)$ is coprime to n ($\gcd(L(g^\lambda(n) \bmod n^2), n) = 1$). As the authority knows what $\lambda(n)$ is it can easily check if a random g satisfies the mentioned property. It chooses $g = 6497955158$. Then:

Voting Authority
$pk = (126869, 6497955158)$
$sk = (293, 433)$

Figure 6.2.: The Paillier keys of the election authority

For the encryption of the plaintext $m = 1010$ Alice has to choose a random number $r \in \mathbb{Z}^*_{16095743161}$. This random number realizes the probabilistic property, where

one plaintext can have many ciphertexts, which will be shown in the next couple of lines.

Encryption	
Alice	**Bob**
$m = 1010$	$m = 1010$
$\mathbf{r = 34}$	$\mathbf{r = 1312}$
$n = 126869,\ \ g = 6497955158,\ \ n^2 = 16095743161$	
$\mathbf{c} = g^m \cdot r^n \bmod n^2$	
$\mathbf{c_A = 4821154392}$	$\mathbf{c_B = 5720727730}$
Decryption	
$m = \frac{L(c^{\lambda(n)} \bmod n^2)}{L(g^{\lambda(n)} \bmod n^2)} \bmod n$ with $\lambda(n) = 31536$	
$L(c_A^\lambda \bmod n^2) = L(15249400063) =$ 120198	$L(c_B^\lambda \bmod n^2) = L(15249400063) =$ 120198
$L(g^\lambda \bmod n^2) = L(6497955158^{31536} \bmod 16095743161) = L(3967320500) = 31271$	
$L(g^\lambda \bmod n^2)^{-1} \bmod n = 53022$	
$\Rightarrow \mathbf{m} = 120198 \cdot 53022 \bmod 126869 = 1010$	

Table 6.3.: The Paillier encryption and decryption of Alice, Bob and the election authority

In the table above there are some calculations which should be stated in more detail:

$$\begin{aligned} c_A = 4821154392 \Rightarrow u_A &= c_A^\lambda \bmod n^2 \\ &= 4821154392^{31536} \bmod 16095743161 = 15249400063 \end{aligned}$$

the same applies to:

$$c_B = 5720727730 \Rightarrow u_B = c_B^{31536} \bmod 16095743161 = 15249400063$$

These c_A, c_B are the input numbers for the function L. Now it is possible to compute

$$L(c_{A,B}^\lambda \bmod n^2) = \frac{u_{A,B} - 1}{n} = \frac{15249400062}{126869} = 120198$$

For decryption we also need to compute $L(g^\lambda \bmod n^2)^{-1}$, therefore we calculate

$$g = 6497955158 \Rightarrow v = g^{31536} \bmod 16095743161 = 3967320500$$
$$L(g^\lambda \bmod n^2) = \frac{v-1}{n} = \frac{3967320499}{126869} = 31271$$

Like in the table above the message is

$$m = \frac{L(c_{A,B}^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n = 120798 \cdot 53022 \bmod 126869.$$

7. Gentry - An Algebraically Homomorphic Scheme

In the decades before Gentry discovered his novel method to gain homomorphic encryption, many researchers worldwide tried to find more powerful and therefore more complex schemes to achieve the fully homomorphic property. Gentry uses a method which no other researcher tried before. Instead of directly creating a superior scheme, he would build one from a "*somewhat*" homomorphic scheme, if its decryption circuit is sufficiently simple. He realized that he could build a fully homomorphic scheme from any scheme that is bootstrappable, i.e., could homomorphically compute a slightly augmented version of its own decryption circuit.

The benefit of fully homomorphic encryption has long been recognized. The question for constructing such a scheme arose within a year of the development of RSA [36]. For more than 30 years, it was unclear whether fully homomorphic encryption was even achievable. During this period, the best encryption system was the Boneh-Goh-Nissim cryptosystem [7] which supports evaluation of an unlimited number of addition operations but one multiplication at the most.

A common reason why a scheme cannot compute circuits of a certain depth is that after a certain amount of computations too much error accumulates, which causes the decryption to obtain a wrong value. The decryption usually is able to handle small amounts of error within a certain range and *bootstrappable encryption* enables "*refreshing*" after some time.

The basic idea of "*refreshing*" is to encrypt under a first key. Compute until right before the error grows too large. Encrypt under a second key. Compute the decryption circuit, which since it stopped before the error grew too large, gives the correct value encrypted under the second key. The first key is no longer required. Continue computation under the second key, and repeat with a new key as often as needed. When the computation has finished, decrypting with the last used key gives the original plaintext.

Gentry's method can be broken down into three major steps:

Step 1: Constructing an encryption scheme using ideal lattices that is **somewhat homomorphic**, which means it is limited to evaluating low-degree polynomials over encrypted data. This scheme is very similar to the Goldreich-

Goldwasser-Halevi scheme published in 1997 [21] which is based on lattice problems as well.

Step 2: "Squashing" the decryption circuit of the original *somewhat homomorphic* scheme to make it *bootstrappable*.

Step 3: Bootstrapping the slightly augmented original scheme of *step 2* to yield the fully homomorphic encryption scheme. This will be done with a "refreshing" procedure.

The innovative idea of Gentry's method of creating a fully homomorphic scheme out of a somewhat homomorphic scheme is the **method of squashing and bootstrapping**. Mathematically the most appealing step is the first step.

7.1. The Somewhat Homomorphic Scheme

The aim of this somewhat homomorphic scheme (SHS) is to construct an encryption scheme that is "almost" **bootstrappable** with respect to a universal set of gates Γ. The reason why ideal lattices are the perfect choice is the fact that the complexity of the decryption algorithms in lattice based encryption schemes are very low, especially compared to schemes like RSA or ElGamal, which rely on exponentiation.

The **key generation** algorithm of Gentry's scheme takes as input a fixed ring R as well as a basis $\mathbf{B}_I$ of a small ideal $I \trianglelefteq R$, which is used to embed the message into an error vector, e.g., $I = (2)$. Additionally an algorithm $\mathsf{IdealGen}(R, \mathbf{B}_I)$ is used to output the *public key* and the *secret key*. The public key consists of a "bad" basis $\mathbf{B}_{pk}$ of the ideal lattice J, e.g., the HNF of the secret key basis B_{sk}. The secret key consists of a "good" basis of J denoted as $\mathbf{B}_{sk}$ with short and nearly orthogonal vectors. The ideal lattice J is chosen such that $I + J = R$, i.e., I and J are relatively prime.
Also remember that if $\vec{v} \in R$ and $\mathbf{B}_J$ is a basis for an ideal $J \trianglelefteq R$, then the value $\vec{v} \bmod \mathbf{B}_J$ is *unique* and can be computed efficiently (see Section 2.1.7 on page 18), i.e., the coset $\vec{v} + J$ has a unique, efficiently-computable *distinguished representative* with respect to the basis B_J. Also $R \bmod \mathbf{B}_J$ denotes the set of distinguished representatives of $\vec{r} + J$ over $\vec{r} \in R$, with respect to the particular basis $\mathbf{B}_J$ of J.

The **encryption** algorithm takes as input the message $\vec{m}$ as well as the public key $\mathbf{B}_{pk}$. The plaintext space P is a subset of $R \bmod \mathbf{B}_I$. It uses an additional algorithm $\mathsf{Samp}(\vec{m}, \mathbf{B}_I)$ to sample a "short" vector from the coset $\vec{m} + I$, this result is further reduced modulo the public basis $\mathbf{B}_{pk}$:

$$\vec{c} = \underbrace{\vec{m} + \vec{i}}_{\vec{e}} \bmod \mathbf{B}_{pk}. \tag{7.1}$$

Geometrically speaking, a ciphertext is a vector $\vec{c}$ close to a J-point, and the encrypted message is encoded in the distance to the nearest lattice point.

The homomorphic property is reflected in the **evaluation** algorithm. Eval takes as input a circuit C of some permitted set $\mathcal{C}_{\mathcal{E}}$ whose gates perform operations modulo $\mathbf{B}_I$, the public key $\mathbf{B}_{pk}$ and a set of input ciphertexts $\Psi = \{\vec{c_1}, ..., \vec{c}_m\}$.
It performs $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ in the proper sequence to compute the output ciphertext $\vec{c}$.

$$\mathsf{Add}(\mathbf{B}_{pk}, \vec{c}_1, \vec{c}_2) \text{ outputs } \vec{c}_1 + \vec{c}_2 \bmod \mathbf{B}_{pk}$$
$$\mathsf{Mult}(\mathbf{B}_{pk}, \vec{c}_1, \vec{c}_2) \text{ outputs } \vec{c}_1 \cdot \vec{c}_2 \bmod \mathbf{B}_{pk}.$$

The reason why this scheme is only somewhat homomorphic is due to the fact that the *error* vectors grows with each operation. Imagine for two ciphertexts $\vec{c_1} = \vec{j_1} + \vec{e_1}$ and $\vec{c_2} = \vec{j_2} + \vec{e_2}$

$$\begin{aligned} \vec{c_+} = \vec{j_+} + \vec{e_+} : \ & \vec{j_+} = \vec{j_1} + \vec{j_2} \in J \\ & \vec{e_+} = \vec{e_1} + \vec{e_2} \text{ is small} \\ \vec{c_\times} = \vec{j_\times} + \vec{e_\times} : \ & \vec{j_\times} = \vec{j_1} \times \vec{j_2} + \vec{j_1} \times \vec{e_2} + \vec{e_1} \times \vec{j_2} \in J \\ & \vec{e_\times} = \vec{e_1} \times \vec{e_2} \text{ is small} \end{aligned}$$

Especially the error originating from multiplications tend to grow very fast, impairing the decryption to decipher incorrectly. This problem will be solved by an additional algorithm that refreshes the ciphertext after each operation. This procedure transforms the so-called somewhat homomorphic scheme into the powerful fully homomorphic scheme.

Now the **decryption** algorithm takes as input the ciphertext and of course the secret key $\mathbf{B}_{sk}$ and outputs the original message

$$\vec{m} = (\vec{c} \bmod \mathbf{B}_{sk}) \bmod \mathbf{B}_I. \tag{7.2}$$

The reason decryption works is that, if the parameters are chosen correctly, then the parallelepiped $\mathcal{P}(B_{sk})$ of the secret key will be a *plump* parallelepiped that contains a sphere of radius bigger than $||\vec{e}||$, so that $\vec{e}$ is indeed the unique point inside $\mathcal{P}(B_{sk})$ that equals $\vec{c} \bmod B_{sk}$. On the other hand, the parallelepiped $\mathcal{P}(B_{pk})$ of the public key will be very *skewed*, and will not contain a sphere of large radius, making it useless for solving *BDDP* (def 2.50.).

Key Generation: $\mathsf{KeyGen}(R, \mathbf{B}_I)$

Input: R and basis $\mathbf{B}_I$ of $I \trianglelefteq R$.

Run $\mathsf{IdealGen}(R, \mathbf{B}_I)$
$(\mathbf{B}_{pk}, \mathbf{B}_{sk}) \leftarrow \mathsf{IdealGen}(R, \mathbf{B}_I)$, where $I + J = R$.
$(\mathbf{B}_{pk}, \mathbf{B}_{sk})$ are bases of J

Output: (pk, sk)
public key: $pk = (R, \mathbf{B}_I, \mathbf{B}_{pk}, \mathsf{Samp})$
secret key: $sk = \mathbf{B}_{sk}$

Encryption: $\mathsf{Enc}(\vec{m}, pk)$

Input: $\vec{m} \in P$ and pk.

Run $\mathsf{Samp}(\vec{m}, \mathbf{B}_I)$
$\vec{e} \leftarrow \mathsf{Samp}(\vec{m}, \mathbf{B}_I)$
Compute $\vec{c} = \vec{e} \bmod \mathbf{B}_{pk}$

Output: $\vec{c} \in R \bmod \mathbf{B}_{pk}$

Evaluation: $\mathsf{Eval}(\mathbf{B}_{pk}, C, \Psi)$

Input: $pk = \mathbf{B}_{pk}$, a circuit $C \in \mathcal{C}_{\mathcal{E}}$ and $\Psi = \{\vec{c}_1, ..., \vec{c}_t\}$

Compute $\vec{c} = g(C)(\Psi) \bmod \mathbf{B}_{pk}$
$g(C)$ denotes the generalized circuit

Output: $\vec{c} \in R \bmod \mathbf{B}_{pk}$

Decryption: $\mathsf{Dec}(\vec{c}, sk)$

Input: $\vec{c} \in R \bmod \mathbf{B}_{pk}$

Compute $\vec{m} = (\vec{c} \bmod \mathbf{B}_{sk}) \bmod \mathbf{B}_I$

Output: $\vec{m} \in P$

Figure 7.1.: The somewhat homomorphic scheme using ideal lattices

7.1.1. Correctness of the SHS

The proof of correctness is split into two parts. Part 1 shows that the decryption works for one ciphertext. Part 2 shows the correctness of the evaluation algorithm.

Decryption

Informally speaking, as already mentioned before, decryption works as long as the secret key $\mathbf{B_{sk}}$ generates a parallelepiped $\mathcal{P}\mathbf{B_{sk}}$ that is plump enough to solve the BDDP in reasonable time. We will later see that this is the case if the columns of ${\mathbf{B_{sk}}}^{-1}$ have Euclidean length smaller than $1/2||\vec{e}||$, where $\vec{e}$ is the error vector $\vec{e} = \vec{m} + \vec{i}$.
A ciphertext in the above described scheme has form $\vec{c} = \vec{e} + \vec{j}$ for some $\vec{j} \in J$. Since $\mathbf{B_{sk}}$ is a basis of the ideal lattice J, $\vec{j} \in J$ can be written as $\vec{j} = \mathbf{B_{sk}} \cdot \vec{\alpha}$ for some integer coefficient vector $\vec{\alpha}$ and therefore

$$\vec{c} = \mathbf{B_{sk}} \cdot \vec{\alpha} + \vec{e}.$$

The decryption procedure needs to reduce $\vec{c} \bmod \mathbf{B_{sk}}$. This is done by computing

$$\begin{aligned} \vec{c} \bmod \mathbf{B_{sk}} &= \vec{c} - \mathbf{B_{sk}} \cdot \left\lfloor {\mathbf{B_{sk}}}^{-1} \cdot \vec{c} \right\rceil = \mathbf{B_{sk}} \cdot \left[{\mathbf{B_{sk}}}^{-1} \cdot \vec{c} \right] \\ &= \mathbf{B_{sk}} \left[{\mathbf{B_{sk}}}^{-1} (\mathbf{B_{sk}} \cdot \vec{\alpha} + \vec{e}) \right] = \mathbf{B_{sk}} \left[\vec{\alpha} + {\mathbf{B_{sk}}}^{-1} \cdot \vec{e} \right] \end{aligned} \tag{7.3}$$

Since the coefficients of $\vec{\alpha}$ are integers and $[\cdot]$ means taking only the fractional part equation (7.3) can be further simplified

$$\mathbf{B_{sk}} \left[\vec{\alpha} + {\mathbf{B_{sk}}}^{-1} \cdot \vec{e} \right] = \mathbf{B_{sk}} \left[{\mathbf{B_{sk}}}^{-1} \cdot \vec{e} \right] \tag{7.4}$$

Remember that we assumed the Euclidean length of the columns of ${\mathbf{B_{sk}}}^{-1}$ to be smaller than $1/2||\vec{e}||$, hence each entry of ${\mathbf{B_{sk}}}^{-1} \cdot \vec{e}$ is smaller than $1/2$ in absolute value, since these entries are all inner products of $\vec{e}$ and columns of ${\mathbf{B_{sk}}}^{-1}$. It follows that the fractional part $\left[{\mathbf{B_{sk}}}^{-1} \cdot \vec{e} \right]$ equals ${\mathbf{B_{sk}}}^{-1} \cdot \vec{e}$ and equation (7.4) simplifies even further

$$\vec{c} \bmod \mathbf{B_{sk}} = \mathbf{B_{sk}} \left[{\mathbf{B_{sk}}}^{-1} \cdot \vec{e} \right] = \mathbf{B_{sk}} \cdot {\mathbf{B_{sk}}}^{-1} \cdot \vec{e} = \vec{e}.$$

From $\vec{e}$ it is easy to extract the original message by reducing it modulo $\mathbf{B}_I$, since $\vec{e} = \vec{m} + \vec{i}$ with $\vec{i} \in I$. So the decryption algorithm is correct if the secret key is chosen correctly.

Evaluation

Considering **correctness** of the evaluation procedure, some structural definitions will be needed. The definitions are directly taken from Gentry's thesis [20]. Before we can state them, the following observation is useful to understand the definitions. The Eval-Algorithm actually uses two different circuits. *First*, it applies a $\pmod{\mathbf{B}_I}$ circuit C to the plaintexts. *Second*, it applies a circuit related to C to the ciphertexts which uses ring operations - not $\pmod{I}$. This circuit is called the **generalized circuit**.

Definition 7.1. (**generalized Circuit**). Let C be a $\pmod{\mathbf{B}_I}$ circuit. The generalized circuit $g(C)$ of C is the circuit formed by replacing C's $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ operations with the ring operations addition $+$ and multiplication $\cdot$ in the ring R.

Definition 7.2. ($\mathbf{X}_{\mathsf{Enc}}$, $\mathbf{X}_{\mathsf{Dec}}$) Let $\mathbf{X}_{\mathsf{Enc}}$ be the image of Samp. Then all ciphertexts output by Enc are in $X_{\mathsf{Enc}} + J$.
Let $\mathbf{X}_{\mathsf{Dec}}$ equal $R \bmod \mathbf{B}_{sk}$, the set of distinguished representatives of cosets of J with respect to the secret basis $\mathbf{B}_{sk}$.

Definition 7.3. (**permitted Circuit**). Let

$$\mathcal{C}_{\mathcal{E}}{}' = \left\{ C \mid \forall (\vec{x}_1, ..., \vec{x}_t) \in X_{\mathsf{Enc}}^t \,:\, g(C)(\vec{x}_1, ..., \vec{x}_t) \in X_{\mathsf{Dec}} \right\}$$

Informally, $\mathcal{C}_{\mathcal{E}}{}'$ is the set of $\pmod{\mathbf{B}_I}$ circuits that, when generalized, the output is always in X_{Dec} if the inputs are in X_{Enc}. That is when the *error* $g(C)(\vec{e_1}, ..., \vec{e_t})$ of the output ciphertexts is small. If $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}}{}'$ then it is called a **set of permitted circuits**.

A different way of looking at those circuits would be that they are permitted as long as the *error*, i.e., $g(C)(\vec{e}_1, ..., \vec{e}_t)$, of the output ciphertext lies inside X_{Dec}, when the input ciphertexts are in the image of Enc which is $X_{\mathsf{Enc}} + J$.

Definition 7.4. (**valid ciphertext**). $\vec{c}$ is called a valid ciphertext with respect to $\mathcal{E}$, the public key pk and permitted circuits $\mathcal{C}_{\mathcal{E}}$ if it equals $\mathsf{Eval}(pk, C, \Psi)$ for some $C \in \mathcal{C}_{\mathcal{E}}$, where each $\vec{c} \in \Psi$ is in the image of Enc. The circuit C may be the identity circuit, in which case the output of Eval is simply an output of Enc.

The following theorem proves correctness of the decryption procedure of a valid ciphertext.

Theorem 7.5. Assume $\mathcal{C}_{\mathcal{E}}$ is a set of permitted circuits containing the identity circuit. Dec correctly decrypts valid ciphertexts $\vec{c}$.

Proof. For a set of ciphertexts $\Psi = \{\vec{c}_1, ..., \vec{c}_t\}$ where a ciphertext is per definition $\vec{c}_k \stackrel{\text{def}}{=} \vec{m}_k + \vec{i}_k + \vec{j}_k$ with $\vec{m}_k \in P$, $\vec{i}_k \in I$, $\vec{j}_k \in J$ and finally $\vec{m}_k + \vec{i}_k \in X_{\mathsf{Enc}}$. When evaluating a circuit C we obtain

$$\vec{c} = \mathsf{Eval}(\mathbf{B}_{pk}, C, \Psi) = g(C)(\Psi) \bmod \mathbf{B}_{pk} \in g(C)(\vec{m}_1 + \vec{i}_1, ..., \vec{m}_t + \vec{i}_t) + J$$

Since C is a permitted circuit, i.e. $C \in \mathcal{C}_{\mathcal{E}}$, and the $(\vec{m}_k + \vec{i}_k)$'s are in X_{Enc} it holds per definition (7.3) that $g(C)(\vec{m}_1 + \vec{i}_1, ..., \vec{m}_t + \vec{i}_t) \in X_{\mathsf{Dec}}$ and therefore in $R \bmod \mathbf{B}_{sk}$. Now decryption proceeds as follows

$$\begin{aligned} \mathsf{Dec}(\vec{c}, \mathbf{B}_{sk}) &= (g(C)(\vec{m}_1 + \vec{i}_1, ..., \vec{m}_t + \vec{i}_t) + J \bmod \mathbf{B}_{sk}) \bmod \mathbf{B}_I \\ &= g(C)(\vec{m}_1 + \vec{i}_1, ..., \vec{m}_t + \vec{i}_t) \bmod \mathbf{B}_I \\ &= C(\vec{m}_1, ..., \vec{m}_t). \end{aligned}$$

Which completes the proof. □

The above theorem proves that the SHS is correct for permitted circuits. Hence the scheme is already fully homomorphic for some set. Now the natural next step is to maximize this set of permitted circuits such that the SHS can evaluate as many polynomials as possible.

7.1.2. Maximizing Circuit Depth

In the section above we saw that the scheme correctly evaluates the gates $\mathsf{Add}_{\mathbf{B}_I}$ and $\mathsf{Mult}_{\mathbf{B}_I}$ if $X_{\mathsf{Enc}} + X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$ and $X_{\mathsf{Enc}} \cdot X_{\mathsf{Enc}} \subseteq X_{\mathsf{Dec}}$ respectively.
In fact due to using ideal lattices these sets have a geometric interpretation. The sets X_{Enc} and X_{Dec} become subsets of $\mathbb{Z}^n$ in the following way:

Definition 7.6. ($\mathbf{r}_{\mathsf{Enc}}$, $\mathbf{r}_{\mathsf{Dec}}$). Let $\mathbf{r}_{\mathsf{Enc}}$ be the *smallest value* such that $X_{\mathsf{Enc}} \subseteq \mathcal{B}(r_{\mathsf{Enc}})$, and
$\mathbf{r}_{\mathsf{Dec}}$ be the *largest value* such that $X_{\mathsf{Dec}} \supseteq \mathcal{B}(r_{\mathsf{Dec}})$.
Now the set of permitted circuits can be defined as

$$\mathcal{C}_{\mathcal{E}} = \{C \,|\, \forall\{\vec{x}_1, ..., \vec{x}_t\} \in \mathcal{B}(r_{\mathsf{Enc}})^t \,:\, g(C)(\vec{x}_1, ..., \vec{x}_t) \in \mathcal{B}(r_{\mathsf{Dec}})\}$$

With this interpretation the question of maximizing the set of permitted circuits becomes a geometric problem. In the following $||\cdot||$ denotes the Euclidean length. In order to maximize the set $\mathcal{C}_{\mathcal{E}}$ it is necessary to bound the Euclidean length $||g(C)(\vec{e}_1, ..., \vec{e}_t)||$. Since $g(C)$ is the generalized circuit this can be done by bounding the length of $||\vec{e_i} + \vec{e_j}||$ and $||\vec{e_i} \times \vec{e_j}||$ in terms of $||\vec{e_i}||$ and $||\vec{e_j}||$.

Theorem 7.7. For a generalized circuit $g(C)$ of a permitted circuit C and lattice vectors $\vec{e}_i$, $i = 1, ..., t$ the Euclidean norm of $||g(C)(\vec{e}_1, ..., \vec{e}_t)||$ can be bound in terms of $||\vec{e_i}||$ and $||\vec{e_j}||$. It holds that

$$||\vec{e_i} + \vec{e_j}|| \leq ||\vec{e_i}|| + ||\vec{e_j}|| \tag{7.5}$$
$$||\vec{e_i} \times \vec{e_j}|| \leq \gamma(R) \cdot ||\vec{e_i}|| \cdot ||\vec{e_j}|| \tag{7.6}$$

$\gamma(R)$ is called the **expansion factor**.

Proof. Equation (7.5): Using the triangle inequality immediately yields the result.
Equation (7.6): Since $\times$ denotes the polynomial multiplication,

$$u(x) \times v(x) = \sum_{k=0}^{n} \sum_{j=0}^{k} u_j v_{k-j} x^{k+j} \;:\; n = \deg(u(x)) + \deg(v(x))$$

it is a bilinear map and its operator norm can be defined as

$$\gamma(R) = \sup_{u,v \neq 0} \frac{||u(x) \times v(x)||}{||u(x)|| \cdot ||v(x)||}$$

which completes the proof. □

Gentry uses this bounds to prove ([20], thm. 7.3.2) that the somewhat homomorphic encryption scheme, described in this section, can correctly evaluate circuits of depth up to

$$\log\log r_{\mathsf{Dec}} - \log\log(\gamma(R) \cdot r_{\mathsf{Enc}}).$$

Hence in order to maximize the depth of circuits that can be correctly evaluated, the expansion factor $\gamma(R)$ and r_{Enc} have to be minimized while r_{Dec} should be maximized.

Example. Since $\gamma(R)$ only depends on the ring itself it directly affects the choice of $f(x)$. There are many possible choices of $f(x)$ for which the expansion factor is only polynomial in n. One where $\gamma(R)$ is particularly small is $f(x) = x^n + 1$.

Let $f(x) = x^n + 1$ and $R = \mathbb{Z}[x]/(x^n + 1)$. Then

$$||\vec{u} \times \vec{v}|| \leq \sqrt{n} \cdot ||\vec{u}|| \cdot ||\vec{v}||$$

For $\vec{u}, \vec{v} \in R$, $\times$ denotes the polynomial multiplication of the unique associated polynomials with coefficient vector $\vec{u}$ resp. $\vec{v}$ in R. Let $z(x) = u(x) \times v(x)$ be the degree $(2n-2)$ product polynomial not yet reduced by $(x^n + 1)$. Then in holds that $z(x) = q(x)f(x) + r(x)$ with $r(x) = z(x) \bmod f(x)$ and $\deg(r(x)) = n - 1$. Then $||\vec{u} \times \vec{v}|| = ||\vec{r}||$ where $||\vec{z}||$ denotes the Euclidean norm of the vector formed by the coefficients of $r(x)$. Since each coefficient of $r(x)$ is an inner product of some coefficients of $u(x)$ and $v(x)$, it holds that $|r_i| \leq ||u|| \cdot ||v||$ (Cauchy-Schwarz inequality).

$$\begin{aligned} ||r(x)||^2 &= |r_0|^2 + ... + |r_{n-1}|^2 \\ &\leq \underbrace{||u(x)||^2 \cdot ||v(x)||^2 + ... + ||u(x)||^2 \cdot ||v(x)||^2}_{\text{n times}} \\ &= n \cdot ||u(x)||^2 \cdot ||v(x)||^2. \end{aligned}$$

This finally yields $||u(x) \times v(x)|| \leq \sqrt{n} \cdot ||u(x)|| \cdot ||v(x)||$. Hence the expansion factor of $R = \mathbb{Z}[x]/(x^n + 1)$ is $\gamma(R) = \sqrt{n}$.

7.1.3. Improving the Decryption Procedure

In order to understand the improvement we have to describe the specific relation between the dual lattice $\mathcal{L}^*$ and the inverse of an ideal I^{-1}. Let J be an ideal lattice in the ring $R = \mathbb{Z}[x]/f(x)$ with $f(x)$ irreducible, recall the following definitions :

$$\begin{aligned} J^* &= \{\vec{x} \in \mathbb{R}^n \mid \forall \vec{v} \in J \ : \ \langle \vec{x}, \vec{v} \rangle \in \mathbb{Z}\} \text{ (dual lattice)}, \\ J^{-1} &= \{\vec{x} \in \mathbb{Q}[x]/f(x) \mid \forall \vec{v} \in J \ : \ \vec{v} \times \vec{x} \in \mathbb{R}\} \text{ (inverse of an ideal)}. \end{aligned}$$

For a $\vec{v} \in R$ the rotation basis V is given by the vectors $\vec{v} \times x^i \bmod f(x)$.

Example. Let $f(x) = x^n + 1$ with n a power of 2. Then $R = \mathbb{Z}[x]/(x^n + 1)$ is closed under *rotation negation*. For

$$\vec{v} = (v_0, ..., v_{n-1}) = v_0 + v_1 x + ... + v_{n-1} x^{n-1} \in R \text{ then,}$$
$$x\vec{v} = x \times \sum_{i=0}^{n-1} v_i x^i = v_0 x + v_1 x^2 + ... + v_{n-2} x^{n-1} - v_{n-1} \in R.$$

Hence the rotation basis of $\vec{v}$ can be written as

$$V = \begin{pmatrix} \vec{v} \\ x\vec{v} \\ \vdots \\ x^{n-1}\vec{v} \end{pmatrix} = \begin{pmatrix} v_0 & v_1 & \cdots & v_{n-1} \\ -v_{n-1} & v_0 & \cdots & v_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ -v_1 & -v_2 & \cdots & v_0 \end{pmatrix}$$

For the next paragraphs let J be a principal ideal generated by $\vec{v}$.

Remark. The ideal $J = (v)$ naturally corresponds to the lattice generated by the rotation basis of $\vec{v}$, denoted as $\mathcal{L}(\mathbf{B_r})$.

Proof. Any $w \in (v)$ has per definition the form $w = a \times_R v$ for some $a \in R$ and $\times_R$ being the ring multiplication. Associating these ring elements with their coefficient vectors yields $w = \sum_i a_i v_i \in \mathcal{L}(\mathbf{B_r})$. Conversely, let $\vec{w} \in \mathcal{L}(\mathbf{B_r})$ then it has the form $\vec{w} = \sum_i a_i v_i$ for some integer a_i. For $a = \sum_i a_i \cdot x^i$ this implies $w = v \times a$ in the ring. □

Let $J = (\vec{v})$ and $\mathbf{B_J}$ denotes the rotation basis of $\vec{v}$, then $J = \mathcal{L}(\mathbf{B_J})$. Then the inverse ideal is generated by $1/\vec{v}$, $J^{-1} = (1/\vec{v})$. Also the rotation basis of $(1/\vec{v})$ is ${\mathbf{B_J}}^{-1}$, hence with the remark from above, $J^{-1} = \mathcal{L}({\mathbf{B_J}}^{-1})$.
The dual of J, denoted J^*, is generated by the inverse transpose of $\mathbf{B_J}$ (theorem 2.35), $J^* = \mathcal{L}((\mathbf{B_J^{-1}})^T)$. Here we see a clear connection between the dual J^* and the inverse J^{-1}, since the bases are just transposes of each other.

Since the product of the rotation basis of some $\vec{v}$ with a vector $\vec{a} \in \mathbb{Q}[x]/f(x)$ can be simply described as $\vec{v} \times \vec{a}$, the decryption function can be simplified to

$$\begin{aligned} \vec{m} = (\vec{c} \bmod \mathbf{B_{sk}}) \bmod \mathbf{B_I} &= \vec{c} - \mathbf{B_{sk}} \cdot \left\lfloor {\mathbf{B_{sk}}}^{-1} \cdot \vec{c} \right\rceil \bmod \mathbf{B_I} \\ &= \vec{c} - \vec{w}_{sk} \times \lfloor \vec{x}_{sk} \times \vec{c} \rceil \bmod \mathbf{B_I}, \end{aligned} \tag{7.7}$$

where $\mathbf{B_{sk}}$ is the rotation basis of some $\vec{w}_{sk} \in \mathbb{Z}[x]/f(x)$, and $\vec{x}_{sk} = 1/\vec{w}_{sk} \in \mathbb{Q}[x]/f(x)$. Since the rotation basis of $\vec{x}_{sk}$ is ${\mathbf{B_{sk}}}^{-1}$ the above holds.

Gentry shows that the equation (7.7) can be further simplified to

$$\vec{m} = \vec{c} - \vec{v}_{sk}^{-1} \times \lfloor \vec{v}_{sk} \times \vec{c} \rceil \bmod \mathbf{B_I} \tag{7.8}$$
$$= \vec{c} - \lfloor \vec{v}_{sk} \times \vec{c} \rceil \bmod \mathbf{B_I} \tag{7.9}$$

where $\mathbf{B_{sk}}$ is a secret key which correctly decrypts for r_{Dec} (see Definition 7.6), the vector is taken from the inverse ideal, $\vec{v}_{sk} \in J^{-1}$, such that the rotation basis of $1/\vec{v}_{sk} = \vec{v}_{sk}^{-1}$ circumscribes a ball of radius r'_{Dec}[1], $\vec{c}$ be a valid ciphertext, i.e., $\vec{c} = \vec{m} + \vec{i} + \vec{j}$ with $\vec{i} \in I$, $\vec{j} \in J$ and $\vec{m} + \vec{i} \in \mathcal{B}(r'_{\mathsf{Dec}})$.
In equation (7.9) Gentry simply institutes a new requirement for the choice of $\vec{v}_{sk}$ in order to make it possible to drop the term $\vec{v}_{sk}^{-1}$. This requirement is that $\vec{v}_{sk} \in J^{-1}$ is also contained in $1 + J^{-1}I$.[2]

Since I and J are relatively prime, there is a vector $\vec{j} \in J \cap (1 + I)$. Let $\vec{r}$ denote the product $\vec{r} = \vec{j} \times \vec{v}_{sk}$, which is in R since $\vec{v}_{sk} \in J^{-1}$. Furthermore for $\vec{v}_{sk} \in 1 + J^{-1}I$ the vector $\vec{r}$ is in $1 + I$. Since decryption is correct for r'_{Dec} it holds that $\vec{v}_{sk}^{-1} \times \lfloor \vec{v}_{sk} \times \vec{c} \rceil \in R$ and therefore

$$\begin{aligned} \vec{v}_{sk}^{-1} \times \lfloor \vec{v}_{sk} \times \vec{c} \rceil &= \vec{r} \times \vec{v}_{sk}^{-1} \times \lfloor \vec{v}_{sk} \times \vec{c} \rceil \bmod I \\ &= \vec{j} \times \lfloor \vec{v}_{sk} \times \vec{c} \rceil \bmod I \\ &= \lfloor \vec{v}_{sk} \times \vec{c} \rceil . \end{aligned}$$

which completes the simplification.

7.1.4. Security of the SHS

The security of the somewhat homomorphic scheme relies on the hardness of the decisional bounded distance decoding problem (**Decision BDDP**). Remember the shortest nonzero vector in a lattice Λ is denoted $\lambda_1(\Lambda)$ (def. 2.36), and the *convex body theorem* (thm. 2.37) gives us an upper bound on the shortest vector of any n-dimensional lattice Λ: $\lambda_1(\Lambda) < \sqrt{n}\det(\Lambda)^{\frac{1}{n}}$.

In the *bounded distance decoding* problem (**BDDP**), one is given a basis $\mathbf{B}$ of some lattice Λ, and a vector $\vec{c}$ that is very close to some Λ-lattice point. The goal is to find the point in Λ nearest to $\vec{c}$.
In the promise problem γ-**BDDP**, there is an additional parameter $\gamma > 1$ and the promise that the target vector $\vec{c}$ is within a distance γ to the nearest lattice point, $\mathrm{dist}(\Lambda, \vec{c}) \leq \sqrt[n]{\det(\Lambda)}/\gamma$.
Extensive experiments with different lattice dimensions and algorithms [16] led researchers to assume that for any k and large n, it takes time 2^k to approximate BDDP in n-dimensional lattices to within a factor of $\alpha = 2^{\mu \cdot n/(k/\log k)}$.

For better understanding of how these security assumptions work the scheme, described in the previous section, the following short description on Babai's rounding off method and its application in the decryption procedure should help.

[1] Gentry showed that the exact value of $r'_{\mathsf{Dec}} = \frac{r_{\mathsf{Dec}}}{n^{2.5} \cdot ||f|| \cdot ||\mathbf{B_I}||}$. I don't want to pursue this issue any further but refer the interested reader to [19, chap. 8.3].
[2] This choice directly affects the value r'_{Dec}.

Babai's Rounding Technique

First of all it should be highlighted that the vector $\vec{v}$ output by the algorithm is not guaranteed to solve γ-BDDP but to approximate it. The idea is rather simple.
Let $\mathbf{B} = \{\vec{b_1}, ...\vec{b_n}\}$ be a basis of the lattice Λ, $\vec{c} \in \mathbb{R}^n$ and $\vec{e} \in \Lambda$ be the nearest lattice point. The idea is to solve the system of linear equations

$$\mathbf{B}\vec{z} = \vec{c}, \quad \vec{z} \in \mathbb{R}^n$$

and then round the coefficients of $\vec{z} = (z_1, ..., z_n)$ to the nearest integer.

$$\vec{v} = \mathbf{B} \lfloor \vec{z} \rceil .$$

This is a possibly good approximation of $\vec{u}$. This procedure can be performed using any basis for the lattice, but Babai proved that $||\vec{c} - \vec{v}||$ is within an exponential factor of the minimal value if the basis is *LLL-reduced* [cf. 28].
So, Babai's Rounding Technique outputs a lattice point $\vec{v}$ such that $\vec{c}-\vec{v} = \sum_{i=1}^{n} m_i \vec{b_i}$, with $|m_i| \leq 1/2$. In other words, the output vector lies within the parallelepiped centered in $\vec{c}$ defined by the basis vectors.

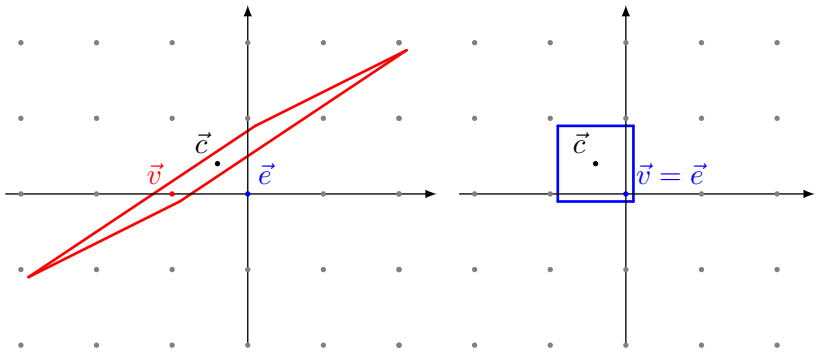

Figure 7.2.: Left: Parallelepiped centered at $\vec{x} = (-0.4, 0.4)$ corresponding to the lattice basis {(3,2);(2,1)}. Right: Parallelepiped centered at $\vec{x} = (-0.4, 0.4)$ corresponding to the lattice basis {(1,0);(0,1)}

Hence having the *good* (nearly orthogonal, and therefore very reduced) secret key basis $\mathbf{B_{sk}}$ allows the key holder to obtain $\vec{e}$ from the ciphertext $\vec{c}$, but with the knowledge of the bad basis $\mathbb{B}_{\neg}$ the adversary cannot retrieve the closest lattice point.

7.1.5. Decryption Complexity

The aim of this initial construction of a somewhat homomorphic encryption scheme was to obtain a scheme that is *bootstrappable*. Up to now we do not know what bootstrappability even means and why it is a necessary prerequisite. This question will

be answered in the following sections. Informally speaking a scheme is bootstrappable if it can homomorphically evaluate its own decryption circuit. Unfortunately this is not the case in this initial scheme [20, chap. 9]. In order to obtain a scheme that can be transformed into a fully homomorphic encryption scheme it is crucial to lower the complexity of the decryption circuit.

7.2. Squashing the Decryption Circuit

Before describing the technique Gentry used to lower the complexity of his initial scheme, I want to recall the scheme with the improvements described in section (7.1.3). Lets denote the "new" scheme $\mathcal{E}^* = (\mathsf{KeyGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*, \mathsf{Eval}^*)$

Key Generation: $\mathsf{KeyGen}^*(R, \mathbf{B}_I)$

Input: R and basis $\mathbf{B}_I$ of $I \trianglelefteq R$.

Run $\mathsf{IdealGen}(R, \mathbf{B}_I)$

$(\mathbf{B}_{pk}, \mathbf{B}_{sk}) \leftarrow \mathsf{IdealGen}(R, \mathbf{B}_I)$

Compute $\vec{v}_{sk} \in J^{-1}$ such that $\mathcal{P}(\mathrm{rot}(\vec{v}_{sk}^{-1})) \supset \mathcal{B}(r_{\mathsf{Dec}}/(2 \cdot n^{2.5}||f|| \cdot ||\mathbf{B_I}||))$

Output: (pk, sk)
public key: $pk = (R, \mathbf{B}_I, \mathbf{B}_{pk}, \mathsf{Samp})$
secret key: $sk = \vec{v}_{sk}$

Encryption: $\mathsf{Enc}^*(\vec{m}, pk) = \mathsf{Enc}(\vec{m}, pk)$

Evaluation: $\mathsf{Eval}^*(\mathbf{B}_{pk}, C, \Psi) = \mathsf{Eval}(\mathbf{B}_{pk}, C, \Psi)$

Decryption: $\mathsf{Dec}^*(\vec{c}, sk)$

Input: $\vec{c} \in R \bmod \mathbf{B}_{pk}$

Compute $\vec{m} = (\vec{c} - \lfloor \vec{v}_{sk} \times \vec{c} \rceil) \bmod \mathbf{B}_I$

Output: $\vec{m} \in P$

Figure 7.3.: The improved somewhat homomorphic scheme

The reason this r'_{Dec} differs from that used in the proof of Equation (7.9) on page 79 by the factor $1/2$, is due to reducing the complexity of the rounding step in Dec^*. It uses $\mathcal{B}(r_{\mathsf{Dec}}/2)$ instead of $\mathcal{B}(r_{\mathsf{Dec}})$ to ensure that the ciphertexts are closer to the

lattice. Although this helps to simplify the decryption circuit, it does not lower it enough to make it bootstrappable.

In order to *squash* the decryption circuit without losing some of its evaluative capacity, Gentry moved some of the decryption computation to the encryption stage, by providing additional information about the secret key in the public key. Of course that by itself weakens the security of the initial scheme.
The transformation works by *splitting* the decryption algorithm Dec^* into two phases.

1. **Phase 1**: An initial computationally intensive preprocessing phase performed by the encrypter without the secret key.
2. **Phase 2**: A computationally lightweight phase performed by the decrypter using the secret key.

The *squashed scheme* therefore introduces three new parameters (τ, r, s) and two new algorithms $\mathsf{SplitKey}, \mathsf{ExpandCT}$ which will be described in detail in the following sections.

7.2.1. The Squashing Transformation

The basic idea behind the transformation is to place a *hint* about the secret key $\vec{v}_{sk}$ inside the public key pk. We will see that this hint is a big set of vectors $\mathcal{S} = \{\vec{t}_i \ : \ i = 1, 2, ..., S\}$ that has a hidden sparse subset τ that adds up to $\vec{v}_{sk}$. In doing so the security of the initial scheme is weakened. To strengthen it a new computationally hard problem is used, namely the **Sparse Subset Sum Problem (SSSP)**.

The scheme $\mathcal{E}^*$(see Figure 7.3) will be altered into a new scheme $\mathcal{E}$ with two additional algorithms, $\mathsf{SplitKey}$, $\mathsf{ExpandCT}$ and the subsequently changed decryption algorithm Dec.

SplitKey

The $\mathsf{SplitKey}$ procedure is used to place the hint about the secret key in the public key. It is a part of the Key Generation procedure KeyGen. The hint consists of a random set of vectors $\tau = \{\vec{t}_1, ..., \vec{t}_r\} \in J^{-1}$ and a *secret* subset of vectors which sum up to the original secret key

$$\vec{v}_{sk^*} = \sum_{i \in S} \vec{t}_i \bmod I.$$

S denotes the distinguished subset of indices $S \subseteq \{1, ..., r\}$ for which the previous equation holds. It takes as input the public and secret key produced by the original

Enc^* and outputs a tuple (sk, τ). The new secret key is a matrix of $0's$ and $1's$ encoding the subset S.

$$sk = \mathbf{SK},\ sk_{ij} = \begin{cases} 1, & \text{iff } j \text{ is the } i^{th} \text{ member of } S \\ 0, & \text{else} \end{cases}$$

τ is added to the original public key yielding the new pk.

ExpandCT

The ExpandCT algorithm is used to prepare the ciphertext for the new *shallower* decryption circuit. It is done by the encrypter himself in the encryption procedure in order to lower the computational complexity of decryption. ExpandCT computes the products

$$\vec{x}_i = \vec{t}_i \times \vec{c} \bmod \mathbf{B_I},\ i = 1, ..., r$$

where $\vec{c}$ is output by the original Enc^* algorithm. The new ciphertext is the tuple

$$\psi = \{\vec{c}; \vec{x}_1, ..., \vec{x}_r\}.$$

Dec

The decryption algorithm Dec takes as input the new secret key **SK** as well as the new ciphertext ψ. With the knowledge of **SK** the decrypter is able to extract the *relevant* $\{\vec{x}_i\}$ and then decrypt using the following equation

$$\vec{m} = \vec{c} - \lfloor \sum_{i \in S} \vec{x}_i \rceil \bmod \mathbf{B_I}. \tag{7.10}$$

Proof. Regarding the extraction of the relevant $\vec{x}_i$'s the decryptor computes a set of vectors $\{\vec{w}_{ij}\}$ with $i = 1, ..., s$ and $j = 1, ..., r$ with

$$\vec{w}_{ij} = sk_{ij} \cdot \vec{c}_j.$$

A vector $\vec{w}_{ij}$ is $\neq \vec{0}$ if and only if $sk_{ij} \neq 0$. This gives us the needed $\vec{x}_i$'s namely the ones for which $i \in S$.
Now remember the original decryption equation (eq. 7.9)

$$\begin{aligned} \vec{m} &= \vec{c} - \lfloor \vec{v}_{sk} \times \vec{c} \rceil \bmod \mathbf{B_I} && (7.9) \\ &= \vec{c} - \lfloor (\sum_{i \in S} \vec{t}_i) \times \vec{c} \rceil \bmod \mathbf{B_I} = \vec{c} - \lfloor \sum_{i \in S} \vec{t}_i \times \vec{c} \rceil \bmod \mathbf{B_I} \\ &= \vec{c} - \lfloor \sum_{i \in S} \vec{x}_i \rceil \bmod \mathbf{B_I}. && (7.10) \end{aligned}$$

□

$(pk, sk) \leftarrow \mathsf{KeyGen}(R, \mathbf{B}_I)$ with $I \trianglelefteq R$

run $\mathsf{IdealGen}(R, \mathbf{B_I})$

$\Rightarrow (\mathbf{B_{pk}}, \mathbf{B_{sk}})$ are two bases of $J \trianglelefteq R$

compute $\vec{v}_{sk} \in J^{-1}$ such that $\mathcal{P}(\mathrm{rot}(\vec{v}_{sk}^{-1})) \supset \mathcal{B}(r_{\mathsf{Dec}}/(2n^{2.5}||f|| \cdot ||\mathbf{B_I}||))$
$pk^* = (R, \mathbf{B_I}, \mathbf{B_{pk}})$
$sk^* = \vec{v}_{sk}$

run $\mathsf{SplitKey}(sk^*, pk^*)$

extract $\vec{v}_{sk^*}$ from sk^*
$\tau = \{\vec{t}_1, ..., \vec{t}_r\}$
$S \subseteq \{1, ..., r\}$ such that $\sum_{i \in S} \vec{t}_i \in \vec{v}_{sk^*} + I$
$\Rightarrow (sk, \tau)$

$pk = (R, \mathbf{B_I}, \mathbf{B_{pk}}, \tau)$
$sk = \mathbf{SK}$ with $sk_{ij} = 1$ iff j is the i^{th} member of S or $sk_{ij} = 0$ else.

$\psi \leftarrow \mathsf{Enc}(\vec{m}, pk)$

run $\mathsf{Samp}(\vec{m}, \mathbf{B_I})$

$\Rightarrow \vec{e} = \vec{m} + i$ with $i \in I$

$\vec{c} = \vec{e} \bmod \mathbf{B}_{pk}$

run $\mathsf{ExpandCT}(\vec{c}, pk)$

$\vec{x}_i = t_i \times \vec{c} \bmod \mathbf{B_I}$ for $i \in [1, r]$.

$\psi = (\vec{c}; \vec{x}_1, ..., \vec{x}_r)$

$\psi \leftarrow \mathsf{Eval}(\mathbf{B}_{pk}, C, \Psi)$
extract $\vec{c}_1$, $\vec{c}_2$ from ψ_1, $\psi_2 \in \Psi$ and perform

$\mathsf{Add}(pk, \vec{c}_1, \vec{c}_2) = \vec{c}_1 + \vec{c}_2 \bmod \mathbf{B}_{pk} = \vec{c}_{1+2}$
$\mathsf{Mult}(pk, \vec{c}_1, \vec{c}_2) = \vec{c}_1 \times \vec{c}_2 \bmod \mathbf{B}_{pk} = \vec{c}_{1\times 2}$

output ψ to include $\vec{c}_{1+2}$ resp $\vec{c}_{1\times 2}$ and the output
$\{\vec{x}_i\} \leftarrow \mathsf{ExpandCT}(\vec{c}_{1+2}, pk)$ resp. $\mathsf{ExpandCT}(\vec{c}_{1\times 2}, pk)$.

$\vec{m} \leftarrow \mathsf{Dec}(\psi, sk)$
set $\vec{w}_{ij} = sk_{ij} \cdot \vec{x}_j$
$\Rightarrow W = \{\vec{w}_{ij}\}$ with $\vec{w}_{ij} \neq \vec{0}$
compute $\vec{m} = \vec{c} - \lfloor \sum_{\vec{x_i} \in W} \vec{x}_i \rceil \bmod \mathbf{B_I}$.

Figure 7.4.: The squashed and improved scheme

In Figure (7.4) we see the resulting scheme, using the above described changes.

Gentry showed that this scheme can homomorphically evaluate its decryption circuit and therefore can be tagged **bootstrappable** [20, chap. 10.3].

7.2.2. Security

As mentioned before, the introduction of τ in the public key requires a second hard computational problem besides the **BDDP** to assure secure communication. This problem will be called the *sparse vector subset sum problem* (**SVSSP**) which is closely related to the well studied *sparse subset sum problem* (**SSSP**), which is known to be **NP**-complete [11].

Definition 7.8 (**SVSSP**)**.** An instance of the SVSSP is a pair $(S, \vec{t})$, where S is a set of vectors $\{\vec{x}_1, ..., \vec{x}_n\}$. The decision problem asks whether there exists a subset T of S that adds up exactly to the target vector $\vec{t}$.

The *search version* of the SVSSP has been studied in the context of *server aided cryptography*. Now it is essential to choose the sets big enough. Obviously if the subset T is *too* small the problem is vulnerable to brute force attacks. Also if $|S|$ is too small, namely if the subset sum solution is unique, then there are numerous lattice reduction attacks similar to those in the field of low-density knapsacks.

Remark. For the interested reader the proof of the reduction

$$\text{SVSSP} \Leftarrow \text{SSSP}$$

can be found in [20, chap. 11].

7.3. The Fully Homomorphic Encryption Scheme

This section will merely outline the final step towards a fully homomorphic encryption scheme, since as mentioned in the introduction of this chapter the first step was the - mathematically speaking - more appealing one. Therefore the following paragraphs are abstract from the scheme described in the previous sections.

It will be shown that *bootstrappability* implies *leveled fully homomorphic* (**LFH**) encryption from which a *fully homomorphic scheme* (**FH**) is derived. The critical property of a *soon-to-be* fully homomorphic encryption scheme $\mathcal{E}$ is the ability to compactly evaluate slight augmentations of its own decryption circuit $D_{\mathcal{E}}$ (*Bootstrappability*).

Notation: For the rest of this chapter following notation is used: a_{i_j} denotes the j^{th} bit of a_i. If a bit a_{i_j} is encrypted under a public key pk_z it is denoted as ${}^{z}[a_{i_j}]$.

7.3.1. Bootstrappability and its Prospects

There is a natural question to this property "*bootstrappability*": Why is this feature so important? Informally speaking, the reason is that bootstrappability allows the scheme to periodically *refresh* ciphertexts to avoid the error growing to vast. Hence, if it is possible to compactly evaluate circuits of depth d, then refresh and evaluate again up to the circuit depth d and so on, it is possible to evaluate circuits of arbitrary depth.

Refreshing could be done by decrypting the ciphertext right before the error grows beyond decryption capability and encrypt it again under a new key. Obviously this would be a major security breach since the plaintext is decrypted at one point in the process. It is desired to refresh without the knowledge of the secret key.

Example. To *refresh* a ciphertext c that encrypts m under a public key pk_1, we *re encrypt* under another public key pk_2, then homomorphically apply the decryption circuit $D_{\mathcal{E}}$ to the result, using the encryption of the first secret key sk_1 under pk_2. Therefore we obtain an encryption of m under pk_2 which can be encrypted with the corresponding secret key sk_2.
Homomorphically applying $D_{\mathcal{E}}$ means decrypting the *inner* ciphertext under pk_1 within homomorphic encryption under pk_2.

More formally, consider a bootstrappable scheme $\mathcal{E}$ with plaintext space P= $\{0, 1\}$. Let (pk_1, sk_1) and (pk_2, sk_2) be two pairs of keys from $\mathcal{E}$. Suppose

$$c_{1(pk_1)} = \mathsf{Enc}(pk_1, m)$$
$${}^{2}[sk_{1_j}] = \mathsf{Enc}(pk_2, [sk_{1_j}]).$$

The vector consisting of all (under pk_2) encrypted secret key bits is denoted by $\overline{{}^2sk_1} = \langle {}^2[sk_{1_1}] \ ... \ {}^2[sk_{1_d}] \rangle$.
Consider the following Recrypt algorithm

Refreshing: $\mathsf{Recrypt}(pk_2, D_{\mathcal{E}}, \overline{{}^2sk_1}, c_{1(pk_1)})$	
Input: The second public key pk_2, the decryption circuit $D_{\mathcal{E}}$, the vector $\overline{{}^2sk_1}$ and the ciphertext c_1 under pk_1.	
Compute	$\overline{{}^2c_1} = \langle {}^2[c_{1_j}] = \mathsf{Enc}(pk_2, c_{1_j}) \rangle_{j \in [1,..,d]}$
	$c_2 = \mathsf{Eval}(pk_2, D_{\mathcal{E}}, \Psi)$ with $\Psi = (\overline{{}^2sk_1}, \overline{{}^2c_1})$
Output: c_2	

Figure 7.5.: Example for the Recrypt algorithm for two ciphertexts.

As long as $\mathcal{E}$ can handle the decryption circuit $D_{\mathcal{E}}$, the output c_2 is obviously an encryption under pk_2 of $\mathsf{Dec}_{\mathcal{E}}(sk_1, c_1) = m$. Hence Recrypt outputs a new encryption of m under pk_2. The fascinating thing about Recrypt is that the Eval algorithm is used to remove the *inner encryption*.

In the context of the initial scheme the Recrypt procedure removes the noise associated to the first ciphertext c_1 under pk_1, due to the fact that decryption removes noise. But Eval simultaneously introduces new noise while evaluating the ciphertexts under pk_2. But as long as the new noise is less than the old noise the *refreshing procedure* has made progress.

This technique is of course useless if it is only applied to the decryption circuit $D_{\mathcal{E}}$. The goal is to perform arbitrary operations on the underlying messages rather than *re-encrypting* the same message.
Suppose $\mathcal{E}$ can handle the decryption circuit $D_{\mathcal{E}}$ augmented by some gate, e.g., Add (see Fig. 4.1). This circuit is denoted D_{Add}. The two ciphertexts,

$$c_{1(pk_1)} = \mathsf{Enc}(m_1, pk_1)$$
$$c_{2(pk_1)} = \mathsf{Enc}(m_2, pk_1)$$

both encrypted under the first public key pk_1, are used to compute

$$\overline{{}^2c_{1(pk_1)}} = \langle {}^2[c_{1(pk_1)_j}] = \mathsf{Enc}(pk_2, c_{1(pk_1)_j}) \rangle_{j \in [1,..,d]}$$
$$\overline{{}^2c_{2(pk_1)}} = \langle {}^2[c_{2(pk_1)_j}] = \mathsf{Enc}(pk_2, c_{2(pk_1)_j}) \rangle_{j \in [1,..,d]},$$

as above. Then it holds that

$$c \leftarrow \mathsf{Eval}(pk_2, D_{\mathsf{Add}}, \overline{{}^2sk_1}, \overline{{}^2c_{1(pk_1)}}, \overline{{}^2c_{2(pk_2)}}) \tag{7.11}$$

is an encryption of $(m_1 + m_2)$ under pk_2.

Example. Imagine an Add gate at level $i + 1$.
It takes as input the encrypted secret key $\overline{{}^{i+1}sk_i}$ and a tuple of ciphertexts associated to output wires at level i that are encrypted under pk_i. The procedure from above homomorphically evaluates D_{Add} to get a ciphertext under pk_{i+1} associated to a wire at level $i + 1$.

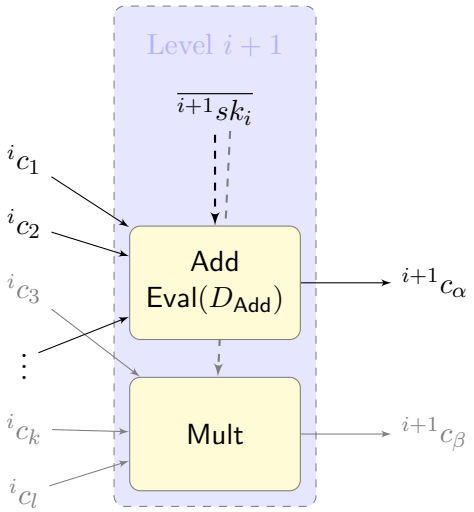

Figure 7.6.: An Add-gate at level $i + 1$

Hence, to be bootstrappable $\mathcal{E}$ has to be able to compactly evaluate not only its decryption circuit but also slightly augmented versions of it.

Definition 7.9. (**Augmented Decryption Circuit**). Let $D_\mathcal{E}$ be $\mathcal{E}$'s *decryption circuit*, which takes a *secret key* sk and *ciphertext* c as input, each formatted as an element of $\mathrm{P}^{l(\lambda)}$[3], where P is the plaintext space. Let Γ be a set of gates with inputs and output in P, which includes the *trivial gate*[4]. A circuit composed of multiple copies of $D_\mathcal{E}$ connected by a single g gate (the number of copies equals the number of inputs to g) is called a **g-augmented decryption circuit**. We denote the set of *g-augmented* decryption circuits with $g \in \Gamma$ by $D_\mathcal{E}(\Gamma)$.

With this notion of an augmented decryption circuit, bootstrappable encryption becomes

Definition 7.10. (**Bootstrappable Encryption**) Let $\mathcal{C}_\mathcal{E}$ denote the set of circuits that $\mathcal{E}$ can *compactly evaluate*. $\mathcal{E}$ is said to be **bootstrappable** with respect to Γ if

$$D_\mathcal{E}(\Gamma) \subseteq \mathcal{C}_\mathcal{E}.$$

[3] l denotes the bit-size of sk.
[4] Meaning input and output are the same.

The above described procedure yields a *leveled fully homomorphic encryption scheme*. The public key in the new scheme $\mathcal{E}^{(d)}$ consists of a sequence of public keys together with a set of encrypted secret keys

$$pk^{(d)} = \left\{pk_0, ..., pk_d; \overline{^1sk_0}, ..., \overline{^dsk_{d-1}}\right\},$$

where sk_i is encrypted under pk_{i+1}. The resulting secret key is $sk^{(d)} = sk_d$. This scheme can be considered as *secure* as the original scheme $\mathcal{E}$ [20, Thm. 4.2.3.].

Unfortunately the key size of the new scheme grows with the depth of the circuit (d). This fact is represented in the following Theorem taken from [20].

Theorem 7.11. If $\mathcal{E}$ is bootstrappable, then, for any integer d, one can construct a scheme $\mathcal{E}^{(d)}$ that can evaluate any circuit of depth d. The decryption circuit for $\mathcal{E}^{(d)}$ is the same as for $\mathcal{E}$, and the complexity of encryption is also the same. $\mathcal{E}^{(d)}$'s public key size is $O(d)$ times that of $\mathcal{E}^{(d)}$'s.

Due to this dependence on d the resulting scheme is called *leveled* rather than fully homomorphic. See the appendix for an instantiation of the described process (Fig. A.1 and Fig. A.2).

7.3.2. From Leveled Fully Homomorphic to Fully Homomorphic

In order to construct a fully homomorphic scheme out of a LFH scheme, it is necessary to lose the *circuit depth dependency*. If it is possible to derive a scheme $\mathcal{E}^{\dagger}$ out of $\mathcal{E}^{(d)}$, for which the public key size is independent of the circuit depth d,then we have achieved our goal.
The obvious way to reduce the public key size is to use only one public key, i.e., for an $\mathcal{E}$ key pair (pk, sk)

$$pk^{\dagger} = \left\{pk; \overline{sk}\right\}$$
$$sk^{\dagger} = sk,$$

where $\overline{sk}$ is the vector containing the encrypted bits of sk under pk.

The new scheme $\mathcal{E}^{\dagger}$ is still *correct* since the Recrypt algorithm works as before, except that the *refreshed* ciphertexts are under the same public key rather than $d + 1$ different ones. Hence, the new scheme uses a *cycle* of encrypted secret keys not an *acyclic chain*.

However, this procedure has a drawback. Changing to a cycle of encrypted secret keys requires another security assumption for $\mathcal{E}$, namely **key-dependent message (KDM)** security, since the security proof [20, Thm. 4.2.3.] breaks when using a *cycle* of secret keys.

This security concept is a rather novel approach. It was first mentioned in 2002 by Black et al. [6], where they constitute that semantic security is not sufficient when dealing with messages that depend on the underlying secret key, which is exactly the case in the above described scheme. Unfortunately in general, a semantically secure encryption scheme is not guaranteed to be *KDM-secure*. On top of that, there is actually no proof that the scheme described so far is, or is not, *KDM-secure*. In the words on Craig Gentry himself:

> *Absent proof of KDM-security in the plain model, one way to obtain fully homomorphic encryption from bootstrappable encryption is simply to assume that the underlying bootstrappable encryption scheme is also KDM-secure. This assumption, though unsatisfying, does not seem completely outlandish. While an encrypted secret key is very useful in a bootstrappable encryption scheme - indeed, one may view this as the essence of bootstrappability - we do not see any actual attack on a bootstrappable encryption scheme that provides a self-encrypted key.*

Although it stays an open question how secure this proposed scheme really is, it is a groundbreaking scientific work for cryptographers all around the world.

Appendix

A. Instantiation of Recrypt

The following two Figures show how a scheme $\mathcal{E}$ that is bootstrappable can be used to construct a leveled fully homomorphic scheme $\mathcal{E}^{(d)}$. The procedure is instantiated recursive.

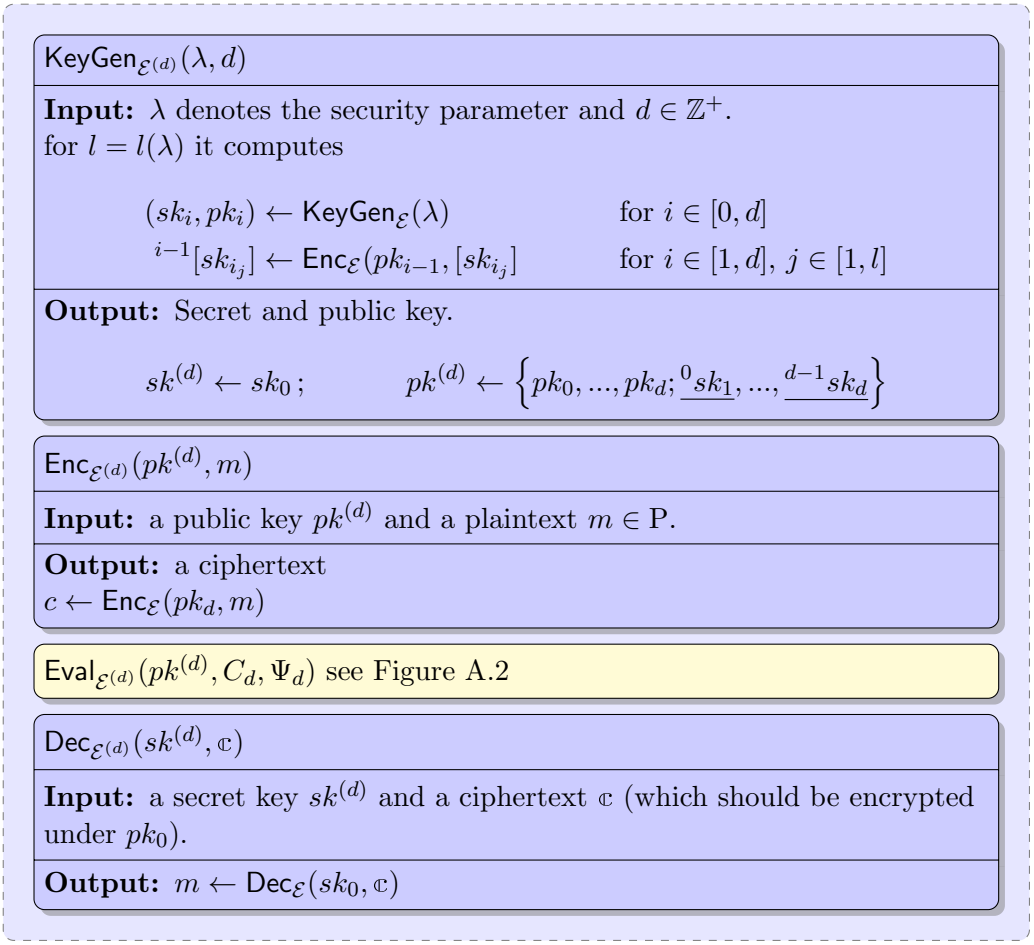

Figure A.1.: From bootstrappable to leveled fully homomorphic.
Part I: $(\mathsf{KeyGen}_{\mathcal{E}^{(d)}}, \mathsf{Enc}_{\mathcal{E}^{(d)}}, \mathsf{Dec}_{\mathcal{E}^{(d)}})$

For the description of the new Eval algorithm lets set $\mathcal{E}^{(\delta)}$ to be the subsystem that

uses $sk^{(\delta)} \leftarrow sk_0$ and $pk^{(\delta)} \leftarrow (\langle pk_i \rangle_{i\in[0,\delta]}, \langle \widehat{sk}_i \rangle_{i\in[1,\delta]})$ for $\delta \leq d$.

$\mathsf{Eval}_{\mathcal{E}^{(\delta)}}(pk^{(\delta)}, C_\delta, \Psi_\delta)$

Input: a public key $pk^{(\delta)}$ and a circuit C_δ of depth δ with gates in Γ and a set of input ciphertexts Ψ_δ where each ciphertext is encrypted under pk_δ.
If $\delta = 0$ it outputs $\Psi_0 = \mathbb{c}$ and terminates, otherwise

set $(C^{\#}_{\delta-1}, \Psi^{\#}_{\delta-1}) \leftarrow \mathsf{Augment}_{\mathcal{E}^{(\delta)}}(pk^{(\delta)}, C_\delta, \Psi_\delta)$

$\mathsf{Augment}_{\mathcal{E}^{(\delta)}}(pk^{(\delta)}, C_\delta, \Psi_\delta)$

Input: the same as in $\mathsf{Eval}_{\mathcal{E}^{(\delta)}}$
$C^{\#}_{\delta-1}$ denotes the augmented circuit $C_\delta + D_{\mathcal{E}}$
$\Psi^{\#}_{\delta-1}$ denotes the set of ciphertexts where each $c_{m_1} \in \Psi_\delta$ is replaced by $(\widehat{sk}_\delta, \widehat{c_j})$, with $\widehat{c_j} \leftarrow \mathsf{Enc}_{\mathcal{E}^{(\delta-1)}}(pk^{(\delta-1)}, c_j)$, and c_j is the j^{th} of c_{m_1}

Output: the tuple $(C^{\#}_{\delta-1}, \Psi^{\#}_{\delta-1})$

set $(C_{\delta-1}, \Psi_{\delta-1}) \leftarrow \mathsf{Reduce}_{\mathcal{E}^{(\delta-1)}}(pk^{(\delta-1)}, C^{\#}_{\delta-1}, \Psi^{\#}_{\delta-1})$

$\mathsf{Reduce}_{\mathcal{E}^{(\delta-1)}}(pk^{(\delta-1)}, C^{\#}_{\delta-1}, \Psi^{\#}_{\delta-1})$

Input: the public key $pk^{(\delta-1)}$ and the tuple $(C_{\delta-1}, \Psi_{\delta-1})$ output by $\mathsf{Augment}_{\mathcal{E}^{(\delta)}}$, with $C_{\delta-1} \in D_{\mathcal{E}}(\Gamma, \delta)$.
Set $C_{\delta-1}$ to be the sub-circuit of $C^{\#}_{\delta-1}$ consisting of the first $\delta - 1$ levels.
Set $\Psi_{\delta-1}$ to be the induced input ciphertexts of $C_{\delta-1}$, where the ciphertext $c^{(w)}_{\delta-1}$ associated to the wire w at level $\delta - 1$ is set to
$c^{(w)}_{\delta-1} \leftarrow \mathsf{Eval}_{\mathcal{E}}(pk_{\delta-1}, C^{(w)}_{\delta-1}, \Psi^{(w)}_{\delta-1})$
and $C^{(w)}_{\delta-1}$ denotes the sub-circuit of $C^{\#}_{\delta-1}$ with output wire w, and $\Psi^{(w)}_{\delta-1}$ are the input ciphertexts.

Output: the tuple $(C_{\delta-1}, \Psi_{\delta-1})$

run $\mathsf{Eval}_{\mathcal{E}^{(\delta-1)}}(pk^{(\delta-1)}, C_{\delta-1}, \Psi_{\delta-1})$

Output: A ciphertext $\mathbb{c}$ which is encrypted under pk_0.

Figure A.2.: From bootstrappable to leveled fully homomorphic. Part 2: $\mathsf{Eval}_{\mathcal{E}^{(\delta)}}$

List of Symbols

2^S	the power set of S
$\mathbb{R}^m$	m-dimensional euclidean space
$\mathbb{Z}_n$	set of congruence classes modulo n
$\mathbb{Z}_n^*$	multiplicative group of $\mathbb{Z}_n$
$R[x]$	set of all polynomials in x with coefficients in the ring R
$\mathcal{B}_m(\vec{0}, r)$	$m-$dimensional open ball of radius r centered in $\vec{0}$
$\mathcal{L}(\mathbf{B})$	lattice generated by the columns of the basis matrix $\mathbf{B}$
$\mathcal{L}^*$	dual lattice
$\mathcal{P}(\mathbf{M})$	half open fundamental parallelepiped to the basis matrix $\mathbf{M}$
$\text{dist}(\mathcal{L}, \vec{t})$	minimum distance from vector $\vec{t}$ to lattice $\mathcal{L}$
Λ	a lattice defined as a nonempty set without any respect to a specific basis
$\gamma(R)$	expansion factor of the ring R
$\lambda_i(\Lambda)$	i^{th} successive minimum of lattice Λ
$\varphi(\cdot)$	Euler totient function
$\lambda(\cdot)$	Carmichael function
$\gcd(a, b)$	greatest common divisor of two non-zero integers a, b
$\text{lcm}(a, b)$	least common multiple of two non-zero integers a, b
$ord_n(a)$	multiplicative order modulo n of an integer a
HNF	Hermite Normal Form
im f	image of a function f
ker f	kernel of a function f
FACT	integer factorization problem
CR	composite residuosity problem
BDDP	bounded distance decoding problem
CVP	closest vector problem
SVP	shortest vector problem

SSSP	sparse subset sum problem
$[a]_\sim$	equivalence class of an element a under a relation $\sim$
$\langle * \rangle$	set generated by the element $*$
$\trianglelefteq$	is used to denote an ideal, $I \trianglelefteq R$
$\cong$	is used to denote the isomorphic relation between two sets
$\lfloor \cdot \rceil$	rounds the coefficients of a vector to the nearest integer
$\sim$	binary relation

List of Tables

List of Figures

Index

Bibliography

[1] ISO/IEC 8859. URL `http://www.iso.org`.

[2] L. Babai. On Lovsz' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[3] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *Proceedings of 28th Annual International Cryptology Conference - CRYPTO 2008*, volume 4622/2007 of *LNCS*, pages 535–552. Springer Berlin / Heidelberg, 2007.

[4] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Proceedings of 28th Annual International Cryptology Conference - CRYPTO 2008*, volume 5157/2008 of *LNCS*, pages 360–378. Springer Berlin / Heidelberg, 2008.

[5] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer Berlin / Heidelberg, 2009.

[6] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75. Springer Berlin / Heidelberg, 2003.

[7] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of Theory of Cryptography (TCC) '05*, LNCS 3378, pages 325–341, 2005.

[8] D. M. Bressoud. *Factorization and Primality Testing*. Springer-Verlag GmbH, Heidelberg, 1989.

[9] D. M. Burton. *Elementary Number Theory*. McGraw-Hill, 6 edition, 2007.

[10] R. D. Carmichael. On composite numbers p which satisfy the fermat congruence $a^{P-1} \equiv 1 \bmod p$. *The American Mathematical Monthly*, 19(2):22–27, 1912.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2nd edition edition, 2001.

[12] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] M. Drmota. *Lecture Notes on Lineare Algebra 1.* Technische Universität Wien, 2005.

[14] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18. Springer-Verlag New York, 1984.

[15] J. Feigenbaum and M. Merritt. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 2, chapter Open Questions, Talk Abstracts, and Summary of Discussions, pages 1–45. ACM, 1991.

[16] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 31–51, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] P. B. Garrett. *Abstract Algebra.* Chapman & Hall/CRC, 2008.

[18] C. F. Gauß. *Disquisitiones Arithmeticae.* Gerhard Fleischer, Lipsiae, 1801.

[19] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[20] C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Department of Computer Science - Stanford University, 2009.

[21] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 112–131. Springer-Verlag, 1997.

[22] S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography.* Massachusetts Institute of Technology, 2008. URL `http://cseweb.ucsd.edu/~mihir/papers/gb.html`.

[23] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–297, 1984.

[24] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC Press, 2008.

[25] R. W. Keener and R. W. Keener. Probability and measure. In *Theoretical Statistics*, Springer Texts in Statistics, pages 1–24. Springer New York, 2010.

[26] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX: 5–83, 1883.

[27] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thom, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. A. Osvik, H. t. Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit rsa modulus. *Cryptology ePrint Archive, Report*, 2010/006:1, 2010.

[28] A. K. Lenstra, H. W. Lenstra, and L. Lovsz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[29] Y.-K. Liu, V. Lyubashevsky, and D. Micciancio. On bounded distance decoding for general lattices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4110 of *Lecture Notes in Computer Science*, pages 450–461. Springer Berlin / Heidelberg, 2006.

[30] A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography*. Discrete Mathematics and Its Applications. CRC Press, Inc., 1996.

[31] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001.

[32] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[33] H. Minkowski. *Geometrie der Zahlen*. Teubner, 1910.

[34] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *ACM Conference on Computer and Communications Security*, pages 59 – 66, 1998.

[35] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology Eurocrypt*, 1592:223–238, 1999.

[36] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *In Foundations of Secure Computation*, pages 169–180, 1978.

[37] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2): 120–126, 1978.

[38] A. Sanjeev and B. Boaz. *Computational Complexity, A Modern Approach*. Cambridge University Press, New York, 2009.

[39] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[40] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 1994.

[41] W. Trappe and L. Washington. *Introduction to Cryptography with Coding Theory.* Prentice Hall, 2nd edition, 2005.