SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

From Zero to One:
A Brief Summary of the Development of Homomorphic Encryption

av

Nick Andersson

2018 - No K15

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

From Zero to One:
A Brief Summary of the Development of Homomorphic Encryption

Nick Andersson

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Jonas Bergstrom

2018

FROM ZERO TO ONE

A Brief Summary of the Development of Homomorphic Encryption

Nick Andersson

March 15, 2018

Abstract

Fully Homomorphic Encryption has been dubbed the Swiss Army knife of cryptography, as it offers a single tool that can be uniformly applied to many cryptographic applications. It allows one to compute arbitrary functions over encrypted data without the decryption key.

This thesis traces the development of Homomorphic Encryption leading up to the first construction of a Fully Homomorphic Scheme by Gentry in 2009. We begin by presenting the basics mathematical foundation as well as a brief treatment of what constitutes a cryptosystem. We then proceed to offer details of Partially Homomorphic Encryption and Somewhat Homomorphic Encryption, both of which are essential pillars of achieving Fully Homomorphic Encryption. Next, we provide an in-depth exposition of Gentrys key result. We end with a treatment of practical applications stemming from the advent of Homomorphic Encryption.

Contents

1 Background

Homomorphism. Just trying to pronounce the word correctly might be a challenge the first time. Even more perplexing might even be the meaning. What does it mean?

As with most terminology in mathematics, a certain degree of awareness of history may enlighten our understanding. In ancient Greece the term $o\mu o\varsigma$ (homo) denoted ”same” while $\mu o\rho\phi\eta$ (morph) denoted ”shape”. The Greek mathematicians were fascinated by these two concepts and would go on to develop a rich theory with regards to them.

Later on, as new mathematical concepts and notions were introduced, the concept homomorphism was coined. It defines a map preserving all the algebraic structures between the domain and range of an algebraic set. This map may simply be a function, which is to say, an operation that takes input from a set of domain and outputs an element in the range. Addition over the real numbers $\mathbb{R}$ and multiplication over the integers $\mathbb{Z}$ are two examples of such operations. To put it more formally:

Definition 1.1. Let G_a and G_b be groups and let $f : G_a \to G_b$ be a function. We say f is a **group homomorphism** if

$$f(a * b) = f(a) * f(b) \tag{1.0.1}$$

for all $a, b \in G_a$.

What we are concerned with in this text is *Homomorphic Encryption* (HE), which is a form of encryption scheme where a third party is able to perform certain computations on encrypted data while preserving the features of the function and format of the encrypted data. For instance, a multiplicative homomorphic encryption scheme, for an encryption function E and the messages m_1 and m_2, one is able to obtain $E(m_1 \cdot m_2)$ by using $E(m_1)$ and $E(m_2)$ **without** knowing m_1 and m_2 explicitly.

There are many practical applications that motivates the study of homomorphic encryption. For instance, it offers a solution to the problem inhibiting many organisations from using cloud computation to analyse and mine data: it continues to pose too much of a security risk to offer a public cloud provider, such as Amazon or Google, access to unencrypted data. Using homomorphic encryption, a company could encrypt its entire database of files (say e-mails) and upload it to the cloud. It could then apply the stored data as it see fit, such as searching the database to understand how its workers collaborate. The results would be downloaded and decrypted **without** exposing the details of a single e-mail.

We may conveniently categorise these various attempts under three main types of schemes, which for the moment we may think of informally as follows:

1. **Partially Homomorphic Encryption (PHE)** allows only one type of operation with an unlimited number of times.

2. **Somewhat Homomorphic Encryption (SHE)** permits some types of operations with a limited number of times.

3. **Fully Homomorphic Encryption (FHE)** allows unlimited number of operations with an unlimited number of times.

In the coming pages, we will look at the historical development of homomorphic encryption schemes, from the advent of PHE, through the development of SHE, before examining in more detail the first construction of FHE. We will then also explore various applications of FHE as well as some of its shortcomings. In short, we will provide an overview that hopefully will satisfy the reader's demand for rigour while also stimulate further inquiry into this exciting new area in cryptography.

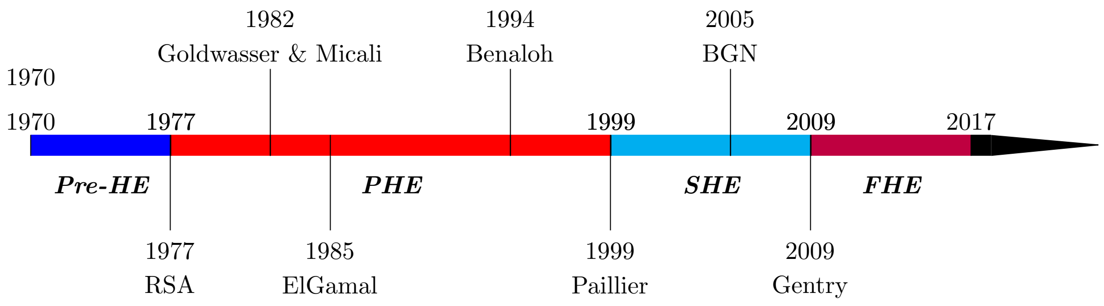


Figure 1. Timeline over the evolution of Homomorphic Encryption systems.

1.1 Outlining Fully Homomorphic Encryption

In principle, a **Fully Homomorphic Encryption** (FHE) permits arbitrary computations on encrypted data. This is to say that if we have some input (plaintext) $m_1, \ldots, m_n$, a function f and $f(m_1, \ldots, m_n)$ it is then possible to compute on encryptions of these inputs $c_1, \ldots, c_n$ obtaining a result which decrypts to $f(m_1, \ldots, m_n)$. For instance, suppose we want to add the integers 24 and 42 in the cloud without revealing the result. What FHE allows is to encrypt $24 \to 37$ and $42 \to 13$, sum these to $37 + 13 = 50$, before then being decrypted to $66 = 24 + 42$.

Most encryption systems have the plaintext (i.e. the input messages) be within some algebraic structure, for instance, a group, ring etc. In those instances, the ciphertext will often also lie in some related structure, possibly the same as that of the plaintext. In the next section, we will provide examples of these "pre-FHE" systems. What unites these schemes is that if the plaintext space is a group G, then the ciphertext space is the product $G \times G$, and f is restricted to the group operation on G. Bearing this in mind, we may view the purpose of FHE to extend the choice of f to be an arbitrary function.

Some might find even the possibility of FHE existing in principle surprising. It may be helpful then to understand fully homomorphic encryption in terms of a physical analogy. Suppose the owner of a toy store (Alice) wants her employees to assemble new toys from raw material yet she fears theft. She seeks to solve this problem by constructing a transparent glove boxes for which only she has the key, and she puts the necessary material inside. By putting on the gloves, an employee can work on the items inside the box. Furthermore, an employee can deposit items into the box - like wood material etc. - even though he/she cannot take anything out. To top it off, the box is transparent, thus allowing the employee to see what he/she is doing. When the employee is done, Alice (alone) is able to recover the now finished product using her own key.

Commentary 1. Note that in our analogy encryption is represented as the employee being unable to remove anything from the box, not that he/she isn't able to see it.

Commentary 2. Evidently this analogy (as is most analogies anyways) inadequate as the glove box might become quite cluttered, whereas in the FHE scheme only the final product need remain.

2 Preliminaries

In this section we will provide a review of the theoretical underpinnings for the homomorphic encryption systems that we will cover later in this thesis. These definitions and result are all from standard textbooks such as (Beachy & Blair, 2006), (Hoffstein, Pipher, Silverman, & Silverman, 2014), (Goldwasser & Bellare, 1996) We will not spend much time on it.The confident reader may skip this part and proceed to the next chapter.

2.1 Groups

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

1. **Associativity:** For all $a, b \in G$ there exists a $c \in G$such that:
$$a \star (b \star c) = (a \star b) \star c$$
2. **Identity:** There is an $e \in G$ such that:
$$e \star a = a \star e$$
for every $a \in G$.
3. **Inverse:** For every $a \in G$ there exists a $a^{-1} \in G$ satisfying:
$$a \star a^{-1} = a^{-1} \star a = e$$

Remark. If $a \star b = b \star a$ for every $a, b \in G$ then the group is said to be **commutative**, or alternatively **abelian**.

Example 2.2. Let

$$G = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{R} \quad \text{and} \quad ad - bc \neq 0 \right\}$$

with operation $\star$ as matrix multiplication. Then this $e = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the inverse is given by the formula:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \begin{pmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{pmatrix}$$

Definition 2.3. Let G be a group, and let H be a subset of G. Then H is called a **subgroup** of G if H is itself a group, under the operation induced by G.

Example 2.4. Let us consider subset of all multiples of a fixed positive integer n in the group $\mathbb{Z}$. In order for us to show that $n\mathbb{Z}$is a subgroup of $\mathbb{Z}$ we must check that each of the requirements of the definition of a group are satisfied.

Let $a, b \in n\mathbb{Z}$. Then we have that $a = n \cdot q$ and $b = n \cdot k$for some $q, k\mathbb{Z}$ and adding gives us $a + b = n \cdot q + n \cdot k = n(q + k)$. From this we see that the sum of two elements in $n\mathbb{Z}$ also belongs to $n\mathbb{Z}$ and so it satisfy the closure requirement.

Next, we note that the associative law holds for all elements in $\mathbb{Z}$, which in means that it in particular also is valid for elements in $n\mathbb{Z}$.

As for the identity element we observe that 0 can be expressed in the form $0 = n \cdot 0$, which means that it also belong to $n\mathbb{Z}$ where it may also serve as identity.

Finally, we consider the inverse of $n\mathbb{Z}$. We observe that $x = n \cdot k$ has the correct form $-x = n \cdot (-k)$ to belong to $n\mathbb{Z}$, and so it also serves as an inverse in $n\mathbb{Z}$.

Definition 2.5. Let G be a group and $a \in G$ be an element of the group. Suppose there exists a positive integer d such that $a^d = e$. The smallest such d is called the **order** of a. If there is no such d, we say that a is of **infinite** order.

Definition 2.6. Let G be a group, and a a random element of G. The set $\langle a \rangle = \{x \in G : x = a^n \quad \text{for some} \quad n \in \mathbb{Z}\}$ is referred to as a **cyclic subgroup** generated by a.

Commentary 3. We say that the group G is a **cyclic group** should there be an element $a \in G$ such that $G = \langle a \rangle$. Moreover, we will call this element a in such an instance the **generator** of G.

Theorem 2.7. *Let G be a finite group. Let H be a subgroup of G. Then the order of H divides the order of G.*

Proof. We have that the left coset of H by $g \in G$ is $gH = \{gh : h \in H\}$, while the right coset of H by g is Hg. The collection of all left cosets of H forms a partition of G, which is to say every element of G is in some left coset of H and all left cosets are pairwise disjoint. The first part of this assertion is easy to show to be true as $x = x \cdot e \in xH$. Turning to the second assertion, suppose $xH \cap yH = \emptyset$ for $x, y \in G$. Then there exists some $h_1, h_2 \in H$ with $xh_1 = yh_2$. Thus if we multiply both sides with h_2^{-1} we have:

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

As H is a group itself $h_1h_2^{-1} \in H$. Ten with $h_1h_2^{-1} = z$

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

Thus, $yH = xH$, as the relationship between x and y is symmetrical. This means the left coset of Hin Gform a partition of G.

Next, we need to show that the order of the left cosets are identical, by demonstrating a bijection from H to xH for any $x \in G$. Let us define the map:

$$
\begin{aligned}
f : H &\to xH \\
g &\mapsto xg
\end{aligned}
$$

If $f(g) = f(g')$, then by definition $xg = xg'$, by multiplying both sides with x^{-1} gives us $g = g'$. What is left to be shown is surjectivity. This is directly seen from the definition of f, as $f(h) = xh$. Thus, all the left cosets of H have the same cardinality as H itself.

Because G is the disjoint union of the left cosets of H, $|H|$ divides $|G|$. $\square$

2.2 Rings

Definition 2.8. Let R be a set on which two binary operations are defined, namely *addition* and *multiplication*, which we denote 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:

1. **Associativity** For all $a, b, c \in R$ we have:
$$
\begin{aligned}
a + (b + c) &= (a + b) + c \\
a \cdot (b \cdot c) &= (a \cdot b) \cdot c
\end{aligned}
$$

2. **Commutative** For all $a, b \in R$ we have:
$$
\begin{aligned}
a + b &= b + a \\
a \cdot b &= b \cdot a
\end{aligned}
$$

3. **Distributive** For all $a, b, c \in R$ we have:
$$
\begin{aligned}
a \cdot (b + c) &= = a \cdot b + a \cdot c \\
(a + b) \cdot c &= a \cdot c + b \cdot c
\end{aligned}
$$

Remark. We say that R is a **commutative ring with identity** if it contains an element 1, assumed to be different from 0, such that $a \in R$, and $a \cdot 1 = 1 \cdot a = 1$.

Definition 2.9. An **integral domain** is a commutative ring with identity such that for any two elements $a, b \in R, a \cdot b = 0$ implies either $a = b$ *or* $b = 0$.

Definition 2.10. A commutative ring in which every element has a multiplicative inverse is called a **field**

Definition 2.11. For a ring $(R, +, \cdot)$, let $(R, +)$ be the underlying additive group. A subset I is called an ideal of R, denoted $I \trianglelefteq R$, if the following conditions are satisfied:

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

Example 2.12. The even integers form an ideal in the ring $\mathbb{Z}$. To see why let $a = 2m$ and $b = 2n$ for $m, n \in \mathbb{Z}$. Then we have that $a \pm b = 2m \pm 2n = 2(m \pm n)$ which is even and so (1) is satisfied. Furthermore we have that any $r \in \mathbb{Z}$ that $ra = 2rm$ which is also even and so (2) is also fulfilled.

We have that the sum as well as the product of two ideals I and J are defined as $\{i + j : i \in I, j \in J\}$ and $\{i \cdot j : i \in I, j \in J\}$.

2.3 Homomorphism

Definition 2.13. A function $f : G \to H$from one group G to another H is said to be a (**group**) homomorphism if the group operation is preserved in the sense

$$f\left(g_1 \star_G g_2\right) = f(g_1) \star_H f(g_2)$$

for all $g_1, g_2 \in G$.

Now, let e_G be the identity in G and e_H the identity in H. We have that a group homomorphism f maps e_G to e_H: $f(e_G) = f(e_H)$.

Definition 2.14. The **kernel** of a group homomorphism $f : G \to H$ is the set of all elements of G which are mapped to the identity element of H.

Example 2.15. Let $f : \mathbb{Z}^2 \to \mathbb{Z}$ be the group homomorphism defined by $f(a, b) = a + b$. Then $(a, b) \in \ker f$ if and only if $f(a, b) = 0$. That is, $(a, b) \in \ker f$ if and only if $a + b = 0$. Hence $(a, b) \in \ker f$ if and only if $b = -a$. As such we have that $\ker f = \{(a, -a) : a \in \mathbb{Z}\}$

Definition 2.16. The **image** of f is like the image of any function, namely:

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

If a group homomorphism $f : G \to H$ is surjective, that is every element in the co-domain is mapped to at least one element in the domain, then H is said to be **homomorphic image** of G. If the group homomorphism $f : G \to H$ gas an inverse homomorphism, then f is said to be an **isomorphism**, furthermore, G and H are said to be **isomorphic**, which we write:

$$G \cong H$$

2.4 Complexity Classes

When dealing with cryptosystems one is often interested in the efficiency of solving that particular system. This is defined as the number of steps that is required to solve an instance of the problem using the most efficient algorithm. To measure time efficiency mathematicians employ a function known as the **Big-O Notation**.

Definition 2.17. Suppose $f(x)$ and $g(x)$ are two functions defined on some subset M of the real numbers $\mathbb{R}$. Then

$$f(x) = \mathcal{O}(g(x))$$

as $x \to \infty$ if and only if there exist a real number x_0 and a positive real number k such that $|f(x)| \leq k \cdot |g(x)|$ for $x \geq x_0$.

The **Big-O Notation** is employed to describe an asymptotic upper bound for a magnitude of a function in terms of another. The next proposition provides a method that one can sometimes use to determine the complexity order of a particular algorithm.

Proposition 2.18. *If the limit*

$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$

exists (and is finite), then $f(x) = \mathcal{O}(g(x))$.

Proof. Let L be the limit. By definition of limit, for any $\epsilon > 0$ there is a constant K, such that

$$\left| \frac{f(x)}{g(x)} - K \right| < \epsilon$$

for all $x > K_\epsilon$. In particular, setting $\epsilon = 1$, we find that

$$\frac{f(x)}{g(x)} < K + 1$$

for all $x > K_1$. Thus, by definition, $f(x) = \mathcal{O}(g(x))$ with $k = K + 1$ and $x_0 > K_1$. $\square$

Definition 2.19. A **polynomial-time algorithm** is an algorithm which runs in polynomial time, which is to say if the number of steps required to complete the algorithm for a given input is $\mathcal{O}(n^k)$

Definition 2.20. 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 $n > \epsilon$ it holds that $f(n) < \frac{1}{p(n)}$

Example 2.21. Let us consider the function $f(n) = 2^{-n}$ and let $c \in \mathbb{N}$ be arbitrary. We may then choose $\epsilon = c^2$. Now for any $n > \epsilon$, we have $2^{-n} = \left(2^{\log_2(n)}\right)^{-\frac{n}{\log_2(n)}} = n^{-\frac{n}{\log_2(n)}}$. Now, as $n > \epsilon$ we know that $\frac{n}{\log_2(n)} > \frac{\epsilon}{\log_2(\epsilon)} > \frac{\epsilon}{\sqrt{\epsilon}} = \sqrt{\epsilon} = c$ (as $\epsilon = c^2$). This implies that $f(n) = 2^{-n} = n^{-\frac{n}{\log_2(n)}} < n^{-c}$ for any $c \in \mathbb{N}$. Thus, it follows that $f(n) = 2^{-n}$ is a negligible function.

Definition 2.22. A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be a **one-way function**, if the following two conditions hold:

1. **Easy to compute**: There exists a polynomial-time function algorithm A computing f, in other words, $A(x) = f(x) = y$ for all x.
2. **Hard to invert**: For every polynomial-time algorithm b, there is a negligible function $v_B(k)$ so that for sufficiently large k:

$$\mathcal{P}\left[B(f(x)) = x\right] \leq v_B(k)$$

Our key takeaway from this is that an one-way function is easy to compute but hard to invert. When we deal with a public key setting these one-way functions are called **trapdoor functions**. This refers to the fact that the key holder has some trapdoor information, which enable him/her to invert the function. As such (1) and (2) holds true for everyone but the key holder.

Remark. It remains an unsolved problem whether there exist any (true) *one-way functions.* So far, no proofs have emerged that show the existence of such functions under reasonable definitions of "easy" and "computationally infeasible".

Remark. Even though it remains unproved whether there exist any *true* one-way functions there are a number of candidates, some of which we will encounter in this text including the discrete logarithm problem, integer factorisation, and the RSA-problem.

The reader may at this point wonder what we mean when we say "easy" and "hard". Isn't it a subjective judgement to say something is "easy"? Well, th *easy* is when the function can be computed by a probabilistic polynomial time algorithm, which is denoted as **PPT**. *Hard*, on the other hand, means that any **PPT** attempting to invert the function will succeed with *negligible* probability.

2.4.1 Deterministic Encryption

A deterministic algorithm will, given a specified input, always return the same output as well as always proceeding in the same manner. It was first introduced into the literature by (Bellare, Boldyreva, & ONeill, 2007).

Input
Algorithm
Output

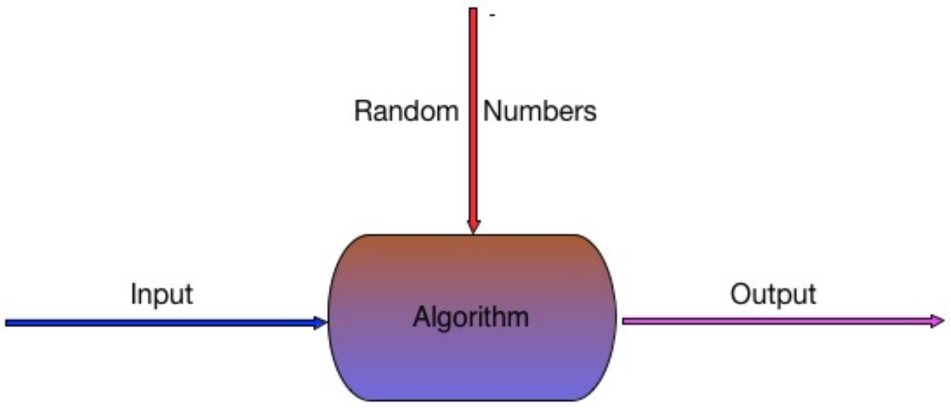
(Goldwasser & Bellare, 1996) brings up three cases of where deterministic encryption is vulnerable:

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 $f(0)$ from $f(1)$.
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 oneway 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.

2.4.2 Probabilistic Encryption

(Goldwasser & Micali, 1984) were first to introduce probabilistic encryption algorithms. The underlying idea is to give an algorithm the ability to generate random numbers [1]. We just saw that there exist certain drawbacks with using deterministic encryption, which fundamentally stems from the fact that a particular plaintext m is paired with a specific cyphertext c_m.

[1]We will not pursue the origins of random numbers as that would take up more space than the entire article



Probabilistic encryption scheme avoid this by employing randomness within the encryption process itself. As a result, there are many possible ciphertexts (say $c_1, \dots, c_r$) for one specific plaintext m. What is remarkable is that if one possesses the right private key, sk, then every possible ciphertext $(c_1, \dots, c_r)$ of a message will be decoded to the original message m.

2.4.3 Defining Public-Key Encryption

Whether we measure the running time of the encryption, decryption, or the adversary algorithms we always use a function of a security parameter k as measurement. This parameter remains fixed from the time the cryptosystem is determined.

We are finally in a position to provide a formal definition of a public-key encryption scheme (Katz & Lindell, 2014).

Definition 2.23. A **public-key encryption scheme** E is a tuple, $(\text{KeyGen}, \text{Enc}, \text{Dec})$ of probablistic polynomial-time algorithms:

1. The **key generation algorithm** (KeyGen) takes the security parameter k as input and outputs a pair of keys (pk, sk). We refer to these as **public key** (pk) and private key (sk).
2. The **encryption elgorithm** (Enc) takes a public-key pk and a string m called the **message** from some underlying message space $\mathbb{M}$ as input. It produces a ciphertext c from an underlying ciphertext space $\mathbb{C}$, which we denote as $Enc(m)$.
3. The **decryption algorithm** (Dec) takes a private-key sk and a ciphertext c as input, and outputs message m.

Remark. In the definition above it is stated that the encryption algorithm is probabilistic. This is not necessarily always the case. Some schemes, mostly those that are a bit older, use encryption algorithms that are deterministic. One such example is the RSA-encryption scheme, which we will encounter later in this text.

2.4.4 Cryptosystems

Definition 2.24. A **cryptosystem** consists of two finite sets M_1 and M_2 together with two functions $E : M_1 \to M_2$ and $D : M_2 \to M_1$ such that:

$$D(E(x)) = x \quad \textbf{and} \quad E(D(y)) = y$$

for all $x \in M_1$ and all $y \in M_2$.

Example 2.25. We begin by providing a very simple example. Let $M_1 = M_2 = \mathbb{Z}_2$ and set $E(x) = R_2(x + 1)$ and $D(y) = R_2(y + 1)$. We are now in a position to send two distinct messages: 0 (”no”) and 1 (”yes”). Let us assume we want to reply ”yes” to a secret question. We encrypt our answer and obtain:

$$E(1) = R_2(1 + 1) = 0$$

Thus, if someone unwarranted received our reply, he/she would obtain 0, which is to say ”no”. For a person who knows our system would decrypt our message:

$$D(E(1)) = R_2(0 + 1) = 1$$

and so this person interprets our message correctly as meaning ”yes”.

While this example illustrates the principle behind cryptosystem, it is too simple to be of any real usage. At best one maybe might be able to use it a couple of times, but then it will almost certainly be broken as the person who tries to listen in will notice the pattern where the counterpart always acts contrary to what our message to him/her is. Let us instead consider another cryptosystem known as the Caesar cipher, named after the famed Field Marshall Julius Caesar, who allegedly used it to communicate with his senior officers during battles.

Example 2.26. As the (English) alphabet consists of 26 letters let $M_1 = M_2 = \mathbb{Z}_{26}$, where we have 0 denote ”a”, 1 ”b” etc. Then we fix an integer $t \in \mathbb{Z}$ and introduce $E_t : \mathbb{Z}_{26} \to \mathbb{Z}_{26}$ as

$$E(x) = R_{26}(x + t)$$

That is to say we transpose each letter by t positions in modulo 26. For instance, if we let $t = 2$ then we have that ”yes” becomes ”agu” and ”no” becomes ”pq”. To decrypt a message we have that decryption function is $D_t : \mathbb{Z}_{26} \to \mathbb{Z}_{26}$

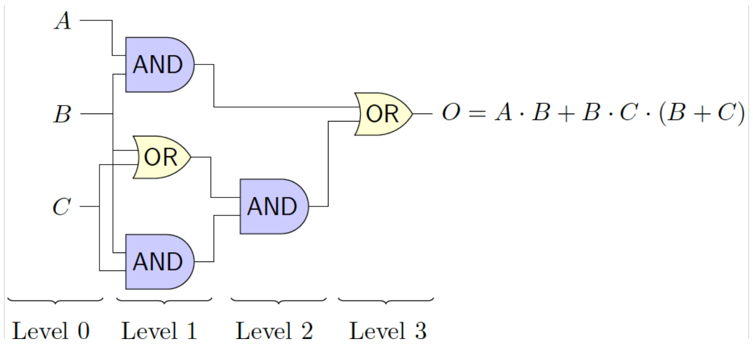
$$D(y) = R_{26}(y - t)$$

While the latter example is more refined compared to the first, it continues to suffer the same weaknesses. One such problem is that should one be in possession of the encryption key then one can also determine the decryption key. This is far from satisfactory: We would like to receive messages from many different users without making it possible for them to read each other's messages. To achieve this, we would need to create different decryption keys for each person, which is very inconvenient.

The Caesar cipher is also easy to break using what is known as **frequency analysis**. That is one observe which letters are used most frequently and substitute those with the most frequent letters in the English (or whatever languages one suspects the text might be written in).

2.4.5 Circuits

Circuits are directed, acyclic graphs. That is to say, it is composed of finitely many vertices and edges, where each edge is directed from vertice to another. As such there is no way to start at any vertex v and move along a directed sequence of edges that will return back to v again. The input values could be integers, boolean values etc. depending on the nature of the circuit. The corresponding gates are set operations and arithmetic operations or logic gates (OR, NOR, AND, NAND,Add, Mult,..).



An example of a circuit representation where the function f outputs the expression $A\dot{B} + B\dot{C}(B + C)$ on input (A, B, C).

We need to define to critical measurements used often in complexity theory, namely **size** and **depth**.

Definition 2.27. The **size** of a circuit C denotes the number of its no-input gates. The **depth** of a circuit C represents the length of its longest path, from an input gate to the output gate, of its underlying directed graph

2.5 Background on Lattices

All of the public key cryptosystems that we have explored so far have either directly or indirectly relied on the difficulty of factoring large numbers or the difficulty of finding discrete logarithms in a finite group. What sets (Gentry et al., 2009) apart is that he relies on a new type of hard problem arising from the study of lattices. A lattice is similar to a vector space, except that instead of being generated by arbitrary real coefficients, all linear combinations of its basis vectors have integer coefficients. While this may appear to be a minor restriction, it actually produces many interesting and subtle questions.

Basing a cryptosystem on lattices offers several benefits over earlier systems, such as faster encryption/decryption and what researchers call quantum resistances. That is to say that there are currently no known quantum algorithms that can swiftly solve hard lattice problems.

Definition 2.28. Let $\vec{v}_1, \dots, \vec{v}_n \in \mathbb{R}^n$ be a set of linearly independent vectors. The **lattice** L **generated by** $\vec{v}_1, \dots, \vec{v}_n$ is the set of linear combinations $\vec{v}_1, \dots, \vec{v}_n$ with coefficients in $\mathbb{Z}$, namely

$$L = \{a_1\vec{v}_1 + \cdots + a_n\vec{v}_n | a_n \in \mathbb{Z}\}$$

Definition 2.29. A **basis** for L is any set of independent vectors that generates L

Theorem 2.30. *Let $L(B), L(B^*)$ be two lattices with $B, B^* \in \mathbb{R}^{n\times n}$ as bases. Then:*

1. *If U is unimodular matrix, then U^{-1} is unimodular.*
2. *$L(B) = L(B*)$ if and only if there exists a unimodular matrix U such that $B^* = BU$.*

Proof. We have that:

1. Given U is unimodular, we have that $U \in \mathbb{Z}^{n\times n}$ **and** $\det(U) = \pm 1$. This means that U is invertible and $\det(U^{-1} = \det(U)^{-1} = \pm 1$. From the identity $U^{-1} = \det(U)^{-1} \cdot \mathrm{adj}(U)$ together with the fact that the entries of $\mathrm{adj}(U)$ are all integers, we deduce $U^{-1} \in \mathbb{Z}^{n\times n}$.
2. $(\rightarrow)$. Assume $L(B) = L(B^*)$ and let $B^* = [b_1^*, \ldots, b_n^*]$. Then $b_1^*, \ldots, b_n^* \in L(B^*) = L(B)$. This means that there exists $U \in \mathbb{Z}^{n\times n}$ such that $B^* = BU$. By the same argument, we have there exists $V \in \mathbb{Z}^{n\times n}$such that $B = B^*V$. Thus, we have $B = BU = B^*VU$. Taking determinants we get $\det(B^*) = \det(B^*) \cdot \det(VU)$. Thus $\det(VU) = 1$ and we have that $\det(U) = \pm 1$.

   $(\leftarrow)$. Suppose now instead that there exists a unimodular matrix Usuch that $B^* = BU$. Writing $B^* = [b_1^*, \ldots, b_n^*]$ we have that $b_1^*, \ldots, b_n^* \in L(B)$ as $U$is an integer matrix. Thus $L(B^*) \subseteq L(B)$. As $B = B^*U^{-1}$ and $U^{-1}$is unimodular, by the same way we get $L(B) \subseteq L(B^*)$. This gives us $L(B) = L(B^*)$.

□

Geometrically we can think of a lattice as an orderly arrangement of points in $\mathbb{R}^m$, where a point is put at the tip of each vector.

Given that a lattice L does not have a unique basis it worthwhile to ask whether or not there is any qualitative difference between two basis. That is to say are some basis "better" than other basis? Actually this turns out not only to be true, but also of fundamental importance in much of the study of lattices as well as the method that Gentry employs to construct a fully homomorphic encryption scheme.

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

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

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

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

Definition 2.33. The **determinant** of a lattice L is the n-dimensional volume of the fundamental parallelpiped $\mathcal{P}(\mathbb{B})$

Definition 2.34. The **dual lattice** of L, denoted L^*, is defined as:

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

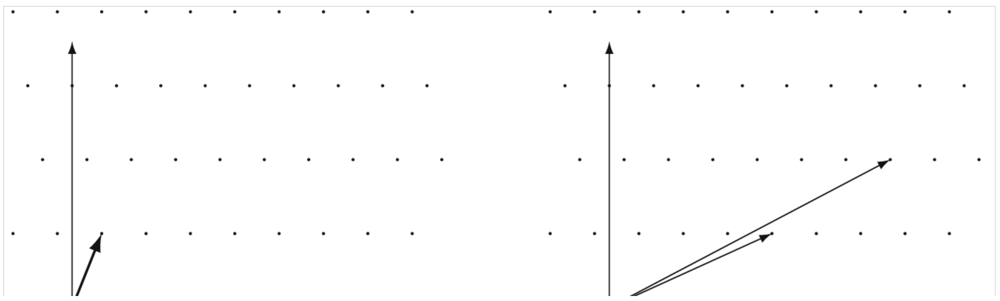
where $\langle \cdot, \cdot \rangle$ denotes the inner product.

If we come to think about the basis vectors $\vec{v}_1, \ldots, \vec{v}_n$ as being vectors of a given length that describe the sides the parallelepiped $\mathcal{P}$, then for basis vectors of given length, the largest volume is obtained when the vectors are pairwise orthogonal to one another. From this we arrive at an important upper bound for the determinant of a lattice known as Hadamard's inequality.

Proposition 2.35. *Let L be a lattice, and take any basis $v_1, \ldots, v_n$ for L. Then*

$$\prod_{i=1}^{n} ||\vec{b}_i|| \geq det(L)$$

We have that as the basis gets closer to be being orthogonal Hadamard's inequality moving closer to being an equality. When we base our cryptography on lattice theory there turns out to be this notion of "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 close to orthogonal. Thus it is fair to conclude that a good basis makes Hadamard's inequality close to an equality.



Two different basis for the same lattice. We say that the first basis is "good", meaning that the vectors are fairly orthogonal, whereas the second basis is "bad" since the angle between the basis vectors is small. The image is from (Hoffstein et al., 2014).

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

$$L_i(L) = \inf\left\{r : \dim\left[\operatorname{span}\left(L \cap \mathcal{B}\left(\vec{0}, r\right)\right)\right] \geq i\right\}$$

where $\mathcal{B}(\vec{0}, r) = \{\vec{x} \in \mathbb{R}^m : ||\vec{x}|| < r\}$ is the m-dimensional open vall of radius r centered in $\vec{0}$.

Theorem 2.37. *For any lattice L of rank n and any convex set $S \subset span(L)$ symmetric about the origin, if $vol(S) > 2^n det(L)$, then S contains a non-zero lattice point $\vec{v} \in S \cap L\{0\}$*

Proof. Define $\widehat{S} = \frac{1}{2}S = \{x : 2x \in S\}$. Then $\operatorname{vol}(\widehat{S}) = 2^{-n}\operatorname{vol}(S) > \det(L)$. By the previous result, there exist two points $z_1, z_2 \in \widehat{S}$ such that $z_1 - z_2 \in L$ is a non-zero lattice point. By definition, $2z_1, 2z_2 \in S$ and because S is centrally symmetric, also $-2z_2 \in S$. Finally as S is convex, $\frac{2z_1 - 2z_2}{2} = z_1 - z_2$ is in S.

□

Definition 2.38. An **ideal lattice** is an integer lattice $L(B) \subseteq \mathbb{Z}^n$ such that $B = \{g \mod f : g \in I\}$ for some monic polynomial f of degree n and ideal $I \subseteq \mathbb{Z}[x]/\langle f \rangle$.

2.6 Problem

2.6.1 The Discrete Logarithm Problem and The Diffie-Hellman Problem

One mathematical problem that arises in many different settings, in this text we will find it underlying the security of ElGamal and Paillier, is the **discrete logarithm problem**. It can be formulated as follows in its most general form.

Definition 2.39. Let G be a group whose group law here will be denoted $\star$. The **Discrete Logarithm Problem** for G is to determine, for any two given elements g and h in G, an integer x satisfying:

$$\underbrace{g \star g \star \cdots \star g}_{x \ times} = h$$

Consider the following issue. You and a friend want to share a private key for usage later, but you only have insecure means of doing this. Each action you might want to take to exchange information can be presumed to be observed by an adversary. How would you go about to share a key without making it available to anyone but your friend? From the outset this would appear to be an impossible task. Nevertheless, (Diffie & Hellman, 1976) managed through a great insight find a way to resolve this problem. The solution is called the **Diffie-Hellman Key Exchange** and works as follows:

1. A trusted party chooses and publishes a (large) prime p, and an integer g having large prime order in $\mathbb{F}_p^*$.
2. Alice chooses a secret integer a and then computes $A \equiv g^a \pmod p$, while Bob chooses another secret integer b, which he then computes $B \equiv g^b \pmod p$.
3. Alice sends A to Bob while he in return sends B to her.
4. Upon receiving B, Alice computes the number $B^a \pmod p$ while Bob computes $A^b \pmod p$. This is now a shared secret key as:

$$B^a \equiv \left(g^b\right)^a \equiv a^{ab} \equiv (g^a)^b \equiv A^b \pmod p$$

The security of Alice's and Bob's newly created public key rests on the difficulty of the following problem.

Definition 2.40. Let p be prime number and g an integer. The *Diffie-Hellman Problem* (**DHP**) is the problem of computing the value of $g^{ab} \pmod p$ from the known values of $g^a \pmod p$ and $g^b \pmod p$.

Definition 2.41. Let $k \in \mathbb{Z}^+$ and let (q_1, q_2, G, G_1, e) be a tuple generated by $\mathcal{G}(k)$, where $n = q_1 \cdot q_2$. Given (n, G, G_1, e) and an element $x \in G$, output "1" if the order of x is q_1 and output "0" otherwise.

Remark. We can rephrase this formulation as follows: Without knowing the factorisation of the group order n, decide if an element x is in a subgroup of G.

Definition 2.42. A number z is said to be a n^{th} if there exists a number $y \in \mathbb{Z}_{n^2}^*$ such that

$$z = y^n \pmod{n^2}$$

2.6.2 Lattice Problems

From a mathematical point of view, our main interest in studying is to determine short vectors in random lattices. Many of the problems encountered in this field can be reduced to two fundamental problems, the shortest vector problem or the closest vector problem.

Definition 2.43. The **shortest vector problem** (SVP) consists of determining a shortest nonzero vector in a lattice L, which is to say find a nonzero vector $v \in L$ that minimises the Euclidean norm $\|v\|$.

Commentary 4. It is worth noting that the shortest vector problem does not have a unique solution. There are instances where there are more than one shortest nonzero vector in lattice. Consider $\mathbb{Z}^2 \subset \mathbb{R}^2$ for example, here all four of the vectors $(0, \pm 1)$ and $(\pm 1, 0)$ are solutions to SVP. This is why we use the X "a" shortest vector and not "the" shortest vector.

Definition 2.44. The **closets vector problem** (CVP) consists of given a vector $\mathbf{w} \in \mathbb{R}^n$ that is not in L, find a vector $v \in L$ that minimises the Euclidean norm $\|w - v\|$.

Commentary 5. Similar to the note given above concerning the non-uniqueness of the solution for the SVP, CVP also lacks a unique solution.

Both SVP and CVP are considered to be "profound problems" (Hoffstein et al., 2014, p. 395), both becoming computationally difficult as the dimension n of the lattice grows.

There is one problem that is related to CVP, called the γ-**bounded distance decoding problem** that deserves special attention. As we shall see later it figures in (?, ?).

Definition 2.45. Given a basis **B** for a lattice L of dimension n and a vector $\vec{t} \in \mathbb{R}^n$ such that $\text{disc}(L, \vec{t}) \cdot \gamma \leq L_1(L)$, find the non-zero vector $\vec{v} \in L$ closests to $\vec{t}$

It has been shown by XX that $\gamma-$**BDDP** is **NP**-hard for any constant factor $\gamma > \frac{1}{\sqrt{2}}$ in general lattice.

3 The Road to Fully Homomorphic Encryption

3.1 Partial Homomorphic Encryption Schemes

In this section we seek to articulate the details of PHE schemes. There are many important examples of PHEs including (Rivest, Shamir, & Adleman, 1978), (Goldwasser & Micali, 1982), (ElGamal, 1985), (Benaloh, 1994), and (Paillier et al., 1999).We choose to focus on the PHE schemes that often serves as basis for other PHE:s, namely RSA-, ElGamal-, and Paillier- encryption.

3.1.1 RSA-Encryption

Any introductory course to cryptography is bound to include extensive treatment of the RSA encryption system (Rivest et al., 1978). Part of its charm is that it uses elements from elementary number theory that are centuries old. To initiate RSA, we multiply two (very large) prime numbers p, q and make their product $n$public. Whereas n is part of the public key (pk), the factors of n are kept secret, where they are part of the secret key (sk). The underlying idea behind is that the factors of n cannot be obtained from n - which is to say the security depends on the difficulty of factoring.

RSA-Procedure

1. **Key Generation** Alice chooses secret primes p and q as well as an encryption exponent e with $\gcd(e,(p-1)(q-1))=1$. In this process she also computes d satisfying $ed \equiv 1 \pmod{(p-1)(q-1)}$.This is her secret key $sk = d$. Alice finally publishes the public key $pk = \{N = pq, e\}$.
2. **Encryption** Bob chooses a plaintext m. Then using Alice's public key $pk = (N, e)$ he computes $c \equiv m^e \pmod{N}$ and sends it to Alice.
3. **Decryption** Upon receiving Bob's encrypted message c, Alice uses her private key $sk = d$ to compute $m' \equiv c^d \pmod{N}$. This m' is then equal to the plaintext m Bob sent.

Theorem 3.1. *For all $x, y \in \mathbb{Z}_N$ we have $D(E(x)) = x$ and $E(D(y)) = y$.*

Proof. We wish to show that $D(E(x)) = x$ which is to say:

$$(x^e)^d \equiv x^{ed} \equiv x \pmod{N}$$

We may rewrite this as:

$$x^{ed} - x \equiv 0 \pmod{N}$$

Thus, for us to prove $D(E(x)) = x$ we have to show n divides the difference $x^{ed} - x$. Since

$$ed = 1 + k(p-1)(q-1) \tag{3.1.1}$$

for some $k \in \mathbb{Z}$. Furthermore, Fermat's little theorem tells us that

$$x^{p-1} \equiv 1 \pmod{p} \tag{3.1.2}$$

Now, we first use 3.1.1 before applying 3.1.2 to obtain:

$$\begin{aligned} x^{ed} &\equiv_p x^{1+k(p-1)(q-1)} \\ &\equiv_p x \cdot \left(x^{p-1}\right)^{k(q-1)} \\ &\equiv_p x \cdot 1^{k(q-1)} \\ &\equiv_p x \end{aligned}$$

This shows p divides $x^{ed} - x$. By switching the p with q we have that q also divides $x^{ed} - x$. Thus it follows that $D(E(x)) = x$.

We do not need to prove the other way, namely $E(D(y)) = y$, since the sets are finite of equal size. □

Example 3.2. Alice chooses $p = 7$ and $q = 13$. She then calculates $N = 7 \cdot 13 = 91$ and $m = 6 \cdot 12 = 72$ before she selects $e = 23$ as $\text{lcm}(e, m) = 1$.

In choosing $e = 23$ Alice also computes d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$ using the Euclidean algorithm.

$$\begin{aligned} 72 &= 3 \cdot 23 + 3 \\ 23 &= 7 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \end{aligned}$$

Reversing the process we have

$$\begin{aligned} 1 &= 3 - 1 \cdot 2 \\ &= 3 - 1 \cdot (23 - 7 \cdot 3) \\ &= 8 \cdot 3 - 1 \cdot 23 \\ &= 8 \cdot (72 - 3 \cdot 23) - 1 \cdot 23 \\ &= 8 \cdot 72 - 25 \cdot 23 \end{aligned}$$

From this Alice can see that $d = 47 \equiv -25 \pmod{72}$. She keeps this as her private key and publish the public key $pk = \{N = 91, e = 23\}$.

At this stage Bob chooses a plaintext $m = 24$. Using the public key $pk = (91, 23)$ that Alice published, he now calculates $c \equiv m^e \pmod{N}$, which in this case may be done by hand using fast-forward algorithm

$$\begin{aligned} 24^{23} &= 24^{16+4+2+1} \\ &= 24^{16} \cdot 24^4 \cdot 24^2 \cdot 24 \end{aligned}$$

and we

$$\begin{aligned} 24^2 &= 576 \equiv 30 \pmod{91} \\ 24^4 = 30^2 &= 900 \equiv 81 \pmod{91} \\ 24^8 &= 81^2 \equiv 9 \pmod{91} \\ 24^{16} &= 9^2 \equiv 81 \pmod{91} \end{aligned}$$

Thus we have $24^{23} = 81 \cdot 81 \cdot 30 \cdot 24 \equiv 19 \pmod{91}$. Bob thus sends the encrypted message 19 to Alice.

Alice is now in a position to determine the message m sent by Bob by using her private key d in computing $m' \equiv_{91} 19^{47}$. Again, this process is done using the fast-forward algorithm, where we rewrite $47 = 24 + 16 + 4 + 2 + 1$. For the sake of brevity we leave it to the reader to find out that Alice indeed will (after some calculations) finds the correct answer, namely $m = 24$.

Homomorphic Property
This system possess an interesting multiplicativity property such that if Alice sends two ciphertexts c_1 and c_2 to Bob where their plaintexts are denoted as m_1 and m_2 then the associated product $m_1 m_2$ is the product of the ciphertexts $c_1 c_2$ since:

$$\begin{aligned} c_1 c_2 &\equiv m_1^e m_2^e \pmod{N} \\ &\equiv (m_1 m_2)^e \pmod{N} \end{aligned}$$

Unfortunately, as $(2+3)^3 \equiv_7 6 \neq 0 \equiv_7 2^3 + 3^3$ demonstrates, this property does not (in general) extend to addition, which is to say:

$$(m_1 + m_2)^e \not\equiv m_1^e + m_2^e \pmod{N}$$

Security

RSA's security is closely tied to the Integer Factorisation Problem that can formally be defined as follows:

Definition 3.3. Let N be a composite integer. The **Integer Factorisation Problem** is to find integer p, where $1 < p < N$, such that p divides N.

There are always various approaches an attacker might apply to break a scheme's security. For instance he/she might try to:

1. Decipher the ciphertext without possessing *sk*.
2. Compute the private key *sk* from the public key key *pk* only.

In the case of RSA, the first approach would be equivalent to the task of computing the e^{th} roots modulo N. This task goes by the name of the **RSA Problem**.

Definition 3.4. Given N, an integer $e > 0$ that is relative prime to $\phi(n)$ and an element $y \in \mathbb{Z}_n^*$. Compute $y^{\frac{1}{e}} \pmod{N}$.

Factoring N and compute the inverse of e remains the most promising approach at this point in time. However, as this is an instance of the *integer factorisation problem* it is currently a computationally hard problem. From the list above, this approach is in line with the second point: to compute *sk* from *pk*.

3.1.2 ElGamal Encryption

While the Diffie-Hellman key exchange algorithm offers a way of publicly sharing a random private key, it is still not a public key cryptosystem since a cryptosystem allows exchange of specific information, not merely a random string of bits. (Rivest et al., 1978) were the first to introduce a public key system and it continues to be a momentous discovery. Nevertheless, even though RSA was historically the first, it was by no means the most natural development of a public key cryptosystem from (Diffie & Hellman, 1976). Instead, that title belongs to (ElGamal, 1985), which we present now.

ElGamal Procedure

1. **Key Generation** Alice generates an efficient description of a cyclic group $\mathbb{Z}_n$ of order p, with generator g. She then chooses a private key $a \in \mathbb{Z}_n$ *randomly* before computing $A = g^a \pmod{p}$. Alice finally publishes A, along with the description of G, q, and g as her public key $pk = \{\mathbb{Z}_n, p, g, A\}$.

2. **Encryption** Bob chooses a plaintext m and a random element k. Using Alice's public key A, he then proceeds to compute $c_1 = g^k \pmod{p}$ and $c_2 = mA^k \pmod{p}$. Bob finally sends these two ciphertexts (c_1, c_2) to Alice.
3. **Decryption** Upon receiving Bob's two ciphertexts, Alice computes $c_2 \cdot (c_1^a)^{-1} \pmod{p}$ which then equals the message m Bob sent.

Theorem 3.5. *For all $x \in \mathbb{Z}_n$ we have $D(E(x)) = x$.*

Proof. We wish to show that $D(E(x)) = x$. As we by assumption already know a, we may compute the quantity

$$x \equiv (c_1^a)^{-1} \pmod{p}$$

Thus can be done by first computing $c_1^1 \pmod{p}$ using the fast power algorithm, before computing the inverse using the extended Euclidean algorithm. We then multiply c_2 by x to obtain:

$$\begin{aligned} x \cdot c_2 &\equiv (c_1^a)^{-1} \cdot c_2 \pmod{p} \\ &\equiv \left(g^{ak}\right)^{-1} \cdot \left(mA^k\right) \pmod{p} \\ &\equiv \left(g^{ak}\right)^{-1} \cdot \left(m\,(g^a)^k\right) \pmod{p} \\ &\equiv m \pmod{p} \end{aligned}$$

□

Remark. ElGamal, unlike RSA, is a probablistic encryption scheme. This means that a single plaintext message m can be encrypted to many possible ciphertexts. It also means that for a given message m, the likelihood of it being encrypted as the same twice is very small. This, as (ElGamal, 1985) notes "prevents attacks like a probable text attack where if the intruder suspects that the plaintext is, for example, m, then he tries to encipher m and finds out if it was really m." The reason as to why this attack, and those similar in nature, will not succeed is due to the fact the sender (Alice) choses a random number a for enciphering. Different values of a will result in different values of $\{c_1, c_2\}$.

Example 3.6. In our example Alice chooses a small prime modulo $p = 457$ and a group generator $g = 266$. Next she selects a random integer $a = 186$, and calculates:

$$\begin{aligned} A &\equiv g^a \pmod{p} \\ &\equiv 266^{186} \pmod{457} \\ &\equiv 257 \pmod{457} \end{aligned}$$

This means that Alice's secret key is $sk = (457, 266, 186)$ and the public key is $pk = (457, 266, 257)$. Upon receiving Alice's public key pk, Bob creates a message $m = 163$ and then selects a random integer $r = 89$. He then calculates the ciphertext (c_1, c_2) where $c_1 = g^r \pmod{p} = 266^{89} \pmod{457} = 443 \pmod{457}$ and $c_2 = m \cdot A^r \pmod{p} = 163 \cdot 257^{347} \pmod{457} = 421 \pmod{457}$

Finally Bob decrypts Alice's message $c_2\left(c_1^g\right)^{-1} \pmod{457} = 421 \cdot \left(443^{266}\right)^{-1} \pmod{457} = 421 \cdot 324^{-1} \pmod{457} = 163 \pmod{457}$, which gives Bob the message $m = 163$, the same Alice sent.

Homomorphic Property
Let p be a prime number, $g \in \mathbb{Z}_p^*$, $a \in \mathbb{Z}_{p-1}$ be a random exponent and $A = g^a \mod p$, and let (pk, sk) be the ElGamal key pair with secret $sk = (p, g, a)$ and public key $pk = (p, g, A)$.

The ElGamal encryption system is homomorphic with respect to the multiplication of plaintexts and ciphertexts. Namely we have that for two messages m_1, m_2 that:

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{a_1}, m_1 \cdot h^{a_1})(g^{a_2}, m_2 \cdot h^{a_2}) \\ &= (g^{a_1} g^{a_2}, m_1 \cdot h^{a_1} m_2 \cdot g^{a_2}) \\ &= \left(g^{a_1+a_2}, m_1 m_2 \cdot h^{a_1+a_2}\right) \\ &= E(m_1 \cdot m_2) \end{aligned}$$

Security
The security of ElGamal depends on the discrete logarithm problem that formally states:

Definition 3.7. Let g be a primitive root for $\mathbb{Z}_p$ and let h be a nonzero element of $\mathbb{Z}_p$. The **Discrete Logarithm Problem** (DLP) is the problem of finding an exponent x such that:

$$g^x \equiv h \pmod{p}$$

Anyone that can compute discrete logarithms is able to get everyone's private key sk and thus break the system. To determine an s such that $g^m = y^r r^s$, on given inputs m and r, is equivalent to the computation of discrete logarithm.

3.1.3 Paillier Encryption

Now that we have seen two examples of multiplicative homomorphic encryption schemes we end this part with an example of an additive homomorphic encryption scheme. In 1999, Pascal Paillier introduced a probablistic public-key algorithm that have come to be known as the Paillier encryption scheme (Paillier et al., 1999). The underlying problem supporting the scheme is the notion that computing nth residue classes is computationally difficult.

Paillier-Procedure

1. **Key Generation**
   Select two large prime numbers p and q such that

   $$\gcd(pq, (p-1)(q-1)) = 1$$

   Denote $n = pq$, and $\lambda = \operatorname{lcm}(p-1, q-1)$. Now choose a random integer g where $g \in \mathbb{Z}_{n^2}^*$. Next check n divides the order of g by investigation the existence of .

   $$\mu = L\left(g^\lambda \pmod{n^2}\right)^{-1} \pmod{n} \tag{3.1.3}$$

where L is defined as:

$$L(u) := \frac{u-1}{n}$$

where the notation $\frac{u-1}{n}$ denotes the quotient, i.e. the largest integer value $t \geq 0$ to satisfy the relation $u-1 \geq t \cdot n$. The public key is $pk = (n, g)$ and the private key is $sk = (\lambda, \mu)$.

2. **Encryption** Select a plaintext $m < n$ and a random $r < n$, then the ciphertext

$$c \equiv g^m \cdot r^n \pmod{n^2} \tag{3.1.4}$$

3. **Decryption** Use the private key $sk = (\lambda, \mu)$ to retrieve the message m by computing:

$$m \equiv L\left(c^\lambda \pmod{n^2}\right) \cdot \mu \pmod{n} \tag{3.1.5}$$

Unlike the previous two examples, proving the correctness of Paillier require some lemmas whose proofs we shall not provide here.

Lemma 3.8. *For any $x \in \mathbb{Z}_n$*

$$(1+n)^x = 1 + xn \pmod{n^2}$$

Proof. We will use induction to prove the lemma. For the base case $m = 0$ the result is evidently clear. Now assume the result holds for $m = x$. We now show it implies that it also holds for $m = x + 1$. We have:

$$\begin{aligned}(1+n)^{x+1} &\equiv (1+xn)(1+n) \pmod{n^2}\\ &\equiv 1 + xn + n \pmod{n^2}\\ &\equiv 1 + (x+1)n \pmod{n^2}\end{aligned}$$

□

Definition 3.9. Let $\mathcal{B} = \{y \in \mathbb{Z}^*_{n^2} | \text{ord}(y) = kn, k \in \{1, \dots, L\}\}$and $g \in \mathcal{B}$. The encryption function E_g is defined as:

$$\begin{aligned}E_g : \mathbb{Z}_n\mathbb{Z}^*_n &\to \mathbb{Z}^*_{n^2}\\ (m, r) &\to g^m r^n \pmod{n^2}\end{aligned}$$

It can be shown that the encryption function E_g is a bijection but rather than spending time proving this result, we note that as E_g is bijective, it follows that it has an inverse. This fact is of usage for the next definition that we introduce.

Definition 3.10. For any $g \in \mathcal{B}$ define the function $[.]_g$ as:

$$\begin{aligned}[.]_g : \mathbb{Z}^*_{n^2} &\to \mathbb{Z}_n\\ c &\mapsto E_g^{-1}(c)[1]\end{aligned}$$

where $E_g^{-1}(c)[1]$ denotes the first component of $E_g^{-1}(c)$.

Lemma 3.11. *For any $c \in \mathbb{Z}^*_{n^2}$ and $g_1, g_2 \in \mathcal{B}$, we have:*

$$[c]_{g_1} \equiv [c]_{g_2}[g_2]_{g_1} \pmod{n}$$
$$[c]_{g_2} \equiv [c]_{g_1}[g_2]^{-1}_{g_1} \pmod{n}$$

Proof. Any number $c \in \mathbb{Z}^*_{n^2}$ can be written in two distinct ways. On the one hand

$$c = \begin{cases} \left(g_1^{[g_2]_{g_1}} r_3^n\right)^{[c]_{g_2}} r_2^n \pmod{n^2} \\ g_1^{[c]_{g_2}[g_2]_{g_1}} \left(r_2 r_3^{[c]_{g_2}}\right)^n \end{cases}$$

At the same time we have

$$c = E_{g_1}\left([c]_{g_1}, r_1\right)$$

By bringing these equations together we have

$$E_{g_1}\left([c]_{g_1}, r_1\right) = E_{g_1}\left([c]_{g_2}[g_2]_{g_1}, r_2 r_3^{[c]_{g_2}}\right) \tag{3.1.6}$$

Now, as E_g is injective (remember that it is in fact bijective) we have that 3.1.6 implies

$$[c]_{g_1} \equiv [c]_{g_2}[g_2]_{g_1} \pmod{n} \tag{3.1.7}$$

With a similar argument one can show that the second part is true. □

Lemma 3.12. *For any $x \in \mathbb{Z}^*_{n^2}$*

$$L\left(x^\lambda \pmod{n^2}\right) \equiv \lambda[x]_{n+1} \pmod{n}$$

Proof. According to lemma 3.8 we have that the order of $n+1$ in $\left(\mathbb{Z}^*_{n^2}, \cdot\right)$ is n. Thus, we have that $n+1 \in \mathcal{B}$. The number $w \in \mathbb{Z}^*_{n^2}$ can be written as

$$w = E_{n+1}\left([w]_{n+1}, y\right) = (n+1)^{[w]_{n+1}} y^n \pmod{n^2}$$

Which implies

$$\begin{aligned} w^L &\equiv (n+1)^{\lambda[w]_{n+1}} y^{n\lambda} \pmod{n^2} \\ &\equiv (n+1)^{\lambda[w]_{n+1}} \pmod{n^2} \\ &\equiv 1 + \lambda[w]_{n+1} n \pmod{n^2} && \text{(By Lemma 3.8)} \end{aligned}$$

Now applying function L we obtain:

$$L\left(w^L \pmod{n^2}\right) \equiv \lambda[w]_{n+1} \pmod{n}$$

□

Theorem 3.13. *For $x \in \mathbb{Z}^*_{n^2}$ we have $D(E(x)) = x$.*

Proof. Given a ciphertext $c \in \mathbb{Z}^*_{n^2}$ for plaintext $m \in \mathbb{Z}$. Then we have that

$$\begin{aligned}
\frac{L\left(c^\lambda \pmod{n^2}\right)}{L\left(g^\lambda \pmod{n^2}\right)} &\equiv \frac{L[c]_{n+1}}{L[g]_{n+1}} && \pmod{n} && \text{(By lemma 3.12)} \\
&\equiv \frac{[c]_{n+1}}{[g]_{n+1}} && \pmod{n} \\
&\equiv [c]_g && \pmod{n} && \text{(By lemma 3.11)} \\
&\equiv m && \pmod{n}
\end{aligned}$$

□

Example 3.14. We illustrate the Paillier encryption scheme with small parameters. Let $p = 7$ and $q = 11$. Then we have $n = pq = 77$. Next, we must select an integer g from $\mathbb{Z}^*_{n^2}$ such that the order of g is a multiple of n in $\mathbb{Z}^*_{n^2}$. By choosing the integer $g = 5652$ we achieve all the necessary properties as the order of g is $2310 = 30 \cdot 77$ in $\mathbb{Z}_{77^2}$.Thus the public key f $(n, g) = (77, 5652)$. The message we wish to encrypt $m = 42$ (the answer to everything) and to do so we choose an integer $r = 23$ such that $r \in \mathbb{Z}_{77}$.

Next we compute the ciphertext:

$$\begin{aligned}
c &= g^m r^n \pmod{n^2} \\
&= 5652^{42} \cdot 23^{77} \pmod{5929} \\
&= 4624 \pmod{5929}
\end{aligned}$$

Now if we wish to decrypt the ciphertext c we first need to compute $\lambda = \text{lcm}(6, 10) = 30$. Define $L(u) = \frac{u-1}{n}$. We have that k is

$$\begin{aligned}
k &= L\left(g^\lambda \pmod{n^2}\right) \\
&= L\left(5652^{30} \pmod{5929}\right) \\
&= L(3928) \\
&= \frac{3928 - 1}{77} \\
&= 51
\end{aligned}$$

We obtain μ by computing the inverse of k in $\pmod{n}$ which gives us:

$$\begin{aligned}
\mu &\equiv k^{-1} \pmod{n} \\
&\equiv 51^{-1} \pmod{77} \\
&\equiv 74 \pmod{77}
\end{aligned}$$

Finally we decrypt the message:

$$\begin{aligned} m &\equiv 74 \cdot L\left(4624^{30} \pmod{5929}\right) \pmod{77} \\ &\equiv 74 \cdot L(4852) \pmod{77} \\ &\equiv 74 \cdot 63 \pmod{77} \\ &\equiv 42 \pmod{77} \end{aligned}$$

which is the same message that we saw encrypted.

Homomorphic Property Given two ciphertexts $E(m_1, pk) = g^{m_1} r_1^n \pmod{n^2}$ and $E(m_2, pk) = g^{m_2} r_2^n \pmod{n^2}$, where r_1, r_2 are randomly chosen from $\mathbb{Z}_n^*$, we have that:

$$\begin{aligned} E(m_1, pk) \cdot E(m_2, pk) &= (g^{m_1} r_1^n)(g^{m_1} r_2^n) \pmod{n^2} \\ &= g^{m_1+m_2}(r_1 r_2)^n \pmod{n^2} \\ &= E(m_1 + m_2, pk) \end{aligned}$$

That is to say the product of two ciphertexts will decrypt to the sum of their corresponding plaintexts, which we may formally express as:

$$D(E(m_1, pk) \cdot E(m_2, pk) \pmod{n^2}) = m_1 + m_2 \pmod{n}$$

Security

The security of the Paillier encryption scheme rests on the composite residuosity assumption, which can be stated as follows:

Definition 3.15. Let $z \in \mathbb{Z}_{n^2}^*$ be a randomly chosen element. It follows from the lemmas presented above that for fixed g, z has a unique representation $z \equiv g^x r^n$, where $x \in \mathbb{Z}_n$ and $r \in \mathbb{Z}_n^*$. The **composite residuosity assumption** says that it is infeasible to compute x from z if the private key sk is not known.

Remark. To compute x from z means to decrypt the Paillier ciphertext z.

This assumption ensures that Paillier encryption is a one-way function. The **decisional composite residuosity assumption**, which is an even stronger assumption, says that it is infeasible to determine whether a randomly chosen element z fro $\mathbb{Z}_{n^2}^*$ is an n^{th} residue.

3.1.4 Summary of PHE

We end this section by summarising the homomorphic properties possess by different PHE schemes.

Name	**Additive**	**Multiplicative**
(Rivest et al., 1978)		$\times$
(Goldwasser & Micali, 1982)	$\times$	
(ElGamal, 1985)		$\times$
(Benaloh, 1994)	$\times$	
(Paillier et al., 1999)	$\times$	

Table 1. Summary of the homomorphic properties possess by various PHE.

3.2 Somewhat Homomorphic Encryption

Now that we have seen various examples of PHEs we proceed to examine more in detail **somewhat homomorphic encryption** (SHE). We may recall our categorisation in the introduction (see chapter 1, introduction) of SHE as: "[A scheme that] permits some types of operations a limited number of times".

3.2.1 Boneh-Goh-Nissim Encryption

2005 marked an important turning point in the study of homomorphic encryption systems. Before then, all cryptosystems' homomorphic properties were restricted to either addition **or** multiplication. Enter (Boneh, Goh, & Nissim, 2005) who using a construct similar to what (Paillier et al., 1999) produced, obtained a system with an additive homomorphism as well as *one* multiplication on encrypted values. As such, BGN represented a signifiant advance towards a fully homomorphic encryption scheme. As we shall see in a moment, Gentry would begin by improving on BGN before using squashing and bootstrapping to construct the first FHE.

Let $\mathcal{G}$ define an algorithm that with a given security parameter $k \in \mathbb{Z}^+$ produces a tuple (q_1, q_2, G, G_1, e) where G, G_1 are groups of order $n = q_1 \cdot q_2$ and p $e : G \times G \to G_1$ is a bilinear map [2]. On input k, the algorithm works as follows:

1. Generate two random k bit primes q_1, q_2 and set $n = q_1 \cdot q_2 \in \mathbb{Z}$.
2. Generate a bilinear group G of order n. Let g be a generator of G and $e : G \times G \to G_1$ be the linear map.
3. Output (q_1, q_2, G, G_1, e)

BGN Procedure

1. **Key Generation** Given a security parameter $k \in \mathbb{Z}^+$, run $\mathcal{G}(k)$ (see above) to obtain a tuple (q_1, q_2, G, G_1, e). Let $n = q_1 \cdot q_2$. Pick two random generators $g, u \leftarrow G$ and set $h = u^{q_2}$. Then h is a random generator of the subgroup G of order q_1. The public key *pk* is generated as (n, G, G_1, e, g, h), while the private key *sk* is q_1.

   Output the public key $pk = (n, G, G_1, e, g, h)$ and the private key $sk = q_1$.

2. **Encryption** It is assumed that the message space consists of integers in the set $\{0, 1 \ldots, T\}$ with $T < q_2$. When encrypting in bits, $T = 1$. To encrypt a message m first select a random number $r \in \mathbb{Z}_n$ and then compute

$$C = g^m h^r \in G \tag{3.2.1}$$

   Output C as the ciphertext.

[2] See (Boneh et al., 2005, p.3) for more about the construction of the bilinear map.

3. **Decryption** To decrypt a ciphertext $E(m)$ is done using the private key $sk = q_1$ by observing that
$$C^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m$$
Set $\hat{g} = g^{q_1}$. To recover m it is enough to compute $\log_{\hat{g}}(C^{q_1})$.

Commentary 6. To recover m can be done using Pollard's lambda method and as $0 \leq m \leq T$ the expected time for this is $O(\sqrt{T})$. As decryption in BOH-system takes polynomial time in the size of the message space T it can only be used to encrypt short messages. Furthermore, as a discrete logarithm cannot be computed quickly, the message should be small for decryption to work efficiently.

Commentary 7. The difficulty of this general discrete logarithm problem depends on the representation of the group. For example, consider G to be the cyclic group of order N. If G is represented as the additive group of $\mathbb{Z}_N$, then computing discrete logarithms in G is equivalent to solving the linear equation $ax \equiv b \pmod{N}$, where a, b given integers. This can be done rather straightforward using the extended Euclidean algorithm. If, however, G is represented as a subgroup of the multiplicative group of say elements from $\mathbb{Z}_m$ - where m may be composite or prime. then then problem can be hard.

Homomorphic Properties

It is rather straightforward to see that the system is additively homomorphic. We have

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{m_1} h^{r_1})(g^{m_2} h^{r_2}) h^r \\ &= g^{m_1+m_2} h^{r_1+r_2+r} \\ &= g^{m_1+m_2} h^{r'} \end{aligned}$$

where $r' = r_1 + r_2 + r$ and it can be seen how $m_1 + m_2$ can be recovered from the ciphertext c. As for the homomorphism over multiplication, we need to use g_1 with order n and h_1 with order q_1. Then we also need to set $g_1 = e(g, g), h_1 = e(g, h)$ and $h = g^{\alpha q_2}$. Suppose now that we are provided two ciphertexts $c_1 = g^{m_1} h^{r_1}$ and $c_2 = g^{m_2} h^{r_2}$ both of which are in G. To build an encryption of the product $m_1 \cdot m_2 \pmod{n}$ we need to 1) Select a random $r \in \mathbb{Z}_n$; 2) Set $c = e(c_1, c_2) h_1^r \in G_1$. Then:

$$\begin{aligned} c &= e(c_1, c_2) h_1^r \\ &= e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r \\ &= g_1^{m_1 m_2} h_1^{m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r} \\ &= g_1^{m_1 m_2} g_1^{r'} \end{aligned}$$

It is important to understand that c now is in the group G_1, not G. As such, one cannot do another homomorphic multiplication operation in G_1. Why? Well, there are no pairing from the set G_1 (Boneh et al., 2005, p. 5).

Security

The hardness of the BGN rests on what is known as the **subgroup decision problem**, which consist of deciding whether an element is a member of a subgroup G_p of group G of composite order $n = pq$, where p and q are distinct primes. (Boneh et al., 2005, p.5-6) proves that system is semantically secure assuming $\mathcal{G}$ satisfy the the subgroup decision assumption.

They go on to state that under this assumption semantic security also exists for ciphertexts in G_1.The reasoning for this is that if semantic security did not hold in G_1, then, as "one can always translate a ciphertext in G to a ciphertext in G_1 by 'multiplying' by the encryption of 1" (Boneh et al., 2005, p. 6), it would not hold in G either.

4 Fully Homomorphic Encryption

The remarkable property of fully homomorphic encryption is that it allows us to run encrypted programs on encrypted data to produce encrypted output, without ever getting exposed to any third parties unencrypted material. Sounds like magic, does it not? Well, until 2009, it really was magic. Then Gentry showed that it is possible to construct a fully homomorphic encryption system.

He first constructs a somewhat homomorphic encryption scheme, that he then uses something he calls squashing before he finally uses a bootstrapping procedure to make it fully homomorphic. To accomplish this Gentry used mathematics that previously had rarely been used in cryptography, namely lattice theory.

As noted by (Gentry et al., 2009, p. 70), it is not sufficient for a scheme to possess a circuit of low decryption complexity for it to be bootstrappable - it also must be able to evaluate that circuit. The previous schemes all have the drawback that they achieve logarithmic depth by permitting the ciphertext size to grow exponentially with the circuit depth. As the ciphertext grows, the decryption circuit must also grow to handle the larger ciphertexts. In short, as one allows larger and larger ciphertexts, the discrepancy between the evaluation depth and the decryption depth widens.

While general lattices possess an additive structure, ideal lattices also have a multiplicative structure that enable them to evaluate deep circuits.

If, however, a lattice $L \subset \mathbb{R}^n$has a basis $\vec{v}_1, \ldots, \vec{v}_n$consisting of vector that are pairwise orthogonal, which is to say $\vec{v}_i \cdot \vec{v}_j = 0$for $i \neq j$, then it is easy to solve both SVP and CVP. In the case of SVP, we note that the length of any vector then in L is:

$$||\sum_{i=1}^{n} a_i \vec{v}_i||^2 = \sum_{i=1}^{n} a_i^2 ||\vec{v}_i||^2$$

Now, as $a_1, \ldots, a_n \in \mathbb{Z}$ we conclude the shortest non-zero vector(s) in L are simply the shortest vector(s) in the set $\{\pm\vec{v}_1, \ldots, \pm\vec{v}_n\}$.

Next, suppose we desire to determine the vector in L that is the closest to a given vector $\vec{w} \in \mathbb{R}^n$. We first write

$$\vec{w} = \sum_{i=1}^{n} t_i v_i \in \mathbb{R}^n$$

Then for $\vec{v} = \sum_{i=1}^{n} a_i v_i \in L$ we have:

$$||\vec{v} - \vec{w}||^2 = \sum_{i=1}^{n} (a_i - t_i)^2 ||\vec{v}_i||^2$$

From this we can easily see that this equation is minimized by selecting the integer closest to the corresponding t_i - recall a_i must be an integer.

One would be forgiven to think that applying this method on an arbitrary basis of L would work as well. In the case that the vectors in the basis are fairly orthogonal to one another, then the chance are good we might successfully solving CVP. Nevertheless, if the basis vectors are highly non-orthogonal then we are unlikely to solve it using this method.

In explaining why lattices were chosen, Gentry begins by setting up a question, namely: "where do we find encryption schemes that have decryption algorithms with low circuit complexity?" (Gentry et al., 2009). Gentry's original approach was to look for a scheme that "evaluates circuits at least as complex as its (augmented) decryption circuit."

Once this is realised it makes little sense to explore variants of those encryption schemes that rely on Diffie-Hellman problem or factoring, such as RSA, ElGamal, Paillier, Boneh-Gog-Nissim etc. Each of these schemes rely on some operation - exponentiation, Legendre symbol computing, pairing etc - that is not yet known to have circuit complexity that are decidable in polylogarithmic time, which is to say they can be solved in time $O(\log^c(n))$ using $O(n^k)$ parallel processors.

The dominant decryption operation for scheme based on lattices, on the other hand, typically are inner product or matrix-vector multiplication. These are operations that are both decidable in polylogarithmic time.

The justification for this is that the CVP and SVP problems are can be solved in polynomial time for the lattices with known good bases. Thus, to retrieve a message from a given ciphertext is equivalent to solve the CVP and SVP problems in polynomial (read practical) time.

4.1 Gentry's Fully Homomorphic Encryption Scheme

Gentry's starting point was somewhat homomorphic encryption scheme based on ideal lattices. Let us recall that this kind of scheme evaluates the ciphertext for a limited number of operations only. Once a certain threshold is reached, the decryption function is unable to recover the message sent correctly. Gentry's stroke of brilliance was to develop two methods called squashing and bootstrapping that enable one to reduce the noise gather during the execution of a homomorphic operation so as to allow the process to continue (Gentry, 2009) (Gentry et al., 2009).

1. Using ideal lattices construct an encryption system that is somewhat homomorphic. This means it can only evaluate low-degree polynomials over encrypted data.
2. Use a technique known as "squashing" on the decryption circuit of the original somewhat homomorphic scheme to make it bootstrappable.
3. Bootstrapping the augmented original scheme to make it a fully homomorphic encryption scheme.

Gentry's somewhat homomorphic encryption scheme based on ideals is presented below:

1. **Key Generation** For the given ring R and the basis B_I of the ideal I, $IdealGen(R, B_I)$ generates the pair of $\left(B_J^{sk}, B_J^{pk}\right)$. $IdealGen(\cdot)$ is an algorithm that produces the relatively prime public and the private key bases of the ideal lattice with basis B_I such that $I + J = R$. A $Samp(\cdot)$ algorithm is also used in the key generation to sample from the given coset of the ideal, where a coset is obtained by shifting an ideal by a certain amount. The public key $pk = \left(R; B_I, B_J^{pk}, Samp()\right)$ and the private key $sk = B_J^{sk}$.

2. **Encryption** For randomly selected vectors $\vec{r}$ and $\vec{g}$, using the public key basis B_J^{pk} chosen from one of the "bad" bases of the ideal lattice L, the message $\vec{m} \in \{0,1\}^n$is encrypted by:

$$c := E(\vec{m}) = \vec{m} + \vec{r} \cdot B_I + \vec{g} \cdot B_J^{pk} \tag{4.1.1}$$

   where B_I is the basis of the ideal lattice L.

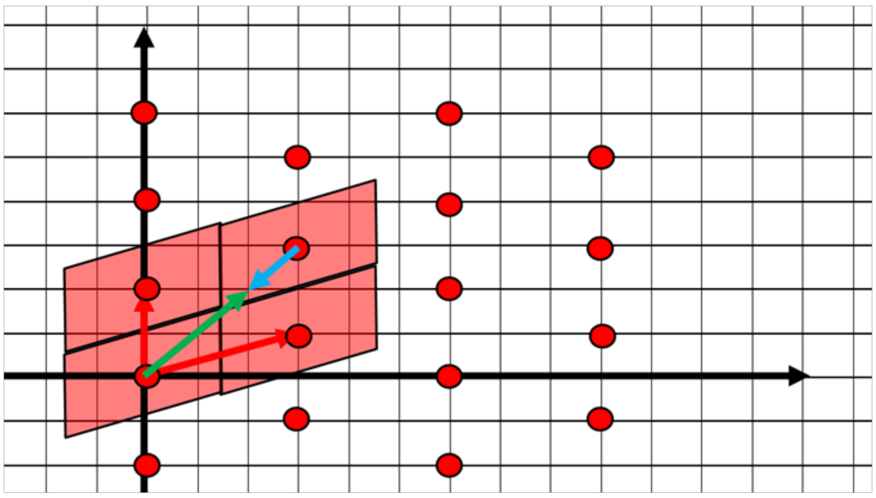
3. **Decryption** Using the secret key basis B_J^{sk} the ciphertext is decrypted as follows:

$$\vec{m} = \vec{c} - B_J^{sk} \cdot \lfloor \left(B_J^{sk}\right)^{-1} \cdot \vec{c}\rceil \mod B_I \tag{4.1.2}$$

   where $\lfloor \cdot \rceil$ is the **nearest integer function** that returns the nearest integers for the coefficients of the vector.

Remark. In the *encryption* phase, we have that the public key consists of a "bad" basis $\mathbf{B}_{pk}$ of the ideal lattice J. More concretely, it is the Hermite Normal Form (see chapter 2.5) of the secret key basis $\mathbf{B}_{sk}$.

Example 4.1. The equation used in the decryption part can be a bit difficult to understand intuitively, which motivates an example. In the diagram below, the red dots are a lattice, and the two red arrows represent a basis, B, for this lattice. We have a green vector, $\vec{c}^* = [2, 2]$. We might ask which point on the lattice $\vec{c}^*$ is closest to. If we are just looking at this in the grid reference frame, $\vec{c}^*$ is equidistant from $(3,1)^T$ and $(3,3)^T$. But within the reference frame of basis $\mathbf{B}$, which is shown by the red parallelograms, $\vec{c}^*$ is closest to $(3,3)^T$. The value of $\mathbf{B} \cdot \left\lfloor \mathbf{B}^{-1}\vec{c}^* \right\rceil$ is the point in the lattice to which $\vec{c}^*$ is closest in the reference frame of $\mathbf{B}$. $\vec{c} = \vec{c}^* - \mathbf{B}\left\lfloor \mathbf{B}^{-1}\vec{c}^* \right\rceil$ is the vector from $\mathbf{B}\left\lfloor \mathbf{B}^{-1}\vec{c}^* \right\rceil$ to the original vector. It could be viewed as the error or remainder when $\vec{c}^*$ is approximated by a point on the lattice. In the diagram below, $\vec{c}$ is the blue arrow.



As the basis vectors are $\vec{b}_1 = (3,1)^T$ and $\vec{b}_2 = (0,2)^T$, we have the basis matrix is:

$$\mathbf{B} = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix}$$

When we actually do the math we find that $(3,3)^T$ is the value of $\mathbf{B}\left\lfloor \mathbf{B}^{-1}\vec{c}^* \right\rceil$ as:

$$\begin{aligned}
\mathbf{B} \cdot \lfloor \mathbf{B}^{-1}\vec{c}^* \rceil &= \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \left\lfloor \frac{1}{6} \begin{pmatrix} 2 & 0 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\rceil \\
&= \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \left\lfloor \begin{pmatrix} \frac{2}{3} \\ \frac{2}{3} \end{pmatrix} \right\rceil \\
&= \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 3 \\ 3 \end{pmatrix}
\end{aligned}$$

As such we have that remainder of $\vec{c}^*$ modulo the basis is:

$$\begin{aligned}
\vec{c} &= \vec{c}^* - B \cdot \lfloor B^{-1}\vec{c}^* \rceil \\
&= \begin{pmatrix} 2 \\ 2 \end{pmatrix} - \begin{pmatrix} 3 \\ 3 \end{pmatrix} \\
&= \begin{pmatrix} -1 \\ -1 \end{pmatrix}
\end{aligned}$$

4.2 Correctness of Somewhat Homomorphic Scheme

4.2.1 Decryption

From a geometrically point of view, we have that decryption function works as long as the secret key $\mathbf{B}_{sk}$ produces a parallelpiped $\mathcal{P}(\mathbf{B}_{sk})$ that is sufficiently plump to solve BDDP in reasonable time. As it turns out, this happens when the columns of $\mathbf{B}_{sk}^{-1}$have Euclidean length smaller than $\frac{1}{2}||\vec{e}||$, where $\vec{e}$ is the error vector $\vec{e} = \vec{m} + \vec{i}$ with $\vec{i} \in I$.

The scheme that we described above produces a ciphertext $\left(\vec{c} = \vec{e} + \vec{j}\right)$ for some $\vec{j} \in J$. Given that $\mathbf{B}_{sk}$ is a basis of the ideal J, we have that $\vec{j} \in J$ may be written as $\vec{j} = \mathbf{B}_{sk} \cdot \alpha$ for some $\alpha \in \mathbb{Z}$, which means that $\vec{c} = \mathbf{B}_{sk} \cdot \alpha\vec{e}$. What the decryption part does is to reduce $\vec{c} \pmod{\mathbf{B}_{sk}}$, which can be done as:

$$\begin{aligned} \vec{c} \pmod{B_{sk}} &= \vec{c} - B_{sk} \cdot \lfloor B_{sk}^{-1} \cdot \vec{c} \rceil \\ &= B_{sk} \cdot [B_{sk}^{-1} \cdot \vec{c}] \\ &= B_{sk}[B_{sk}^{-1} (B_{sk} \cdot \vec{\alpha} + \vec{e})] \\ &= B_{sk}[\vec{\alpha} + B_{sk}^{-1} \cdot \vec{e}] \end{aligned}$$

As the coefficients of $\vec{\alpha}$ are integers and $[\cdot]$ means taking only the fractional part we may simply the last expression further such that:

$$B_{sk}[\vec{\alpha} + B_{sk}^{-1} \cdot \vec{e}] = B_{sk}[B_{sk}^{-1} \cdot \vec{e}]$$

Since we assume that the Euclidean length of the columns B_{sk}^{-1} is smaller than $\frac{1}{2}||\vec{e}||$, we have that each entry of $B_{sk}^{-1} \cdot \vec{e}$ is smaller than $\frac{1}{2}$ in absolute value. This is due the fact all entries are inner products of B_{sk}^{-1} and $\vec{e}$. It follows that fractional part $[B_{sk}^{-1} \cdot \vec{e}]$ is equal to $B_{sk}^{-1} \cdot \vec{e}$ and so

$$\begin{aligned} \vec{e} \pmod{B_{sk}} &= B_{sk}[B_{sk}^{-1} \cdot \vec{e}] \\ &= B_{sk} \cdot B_{sk}^{-1} \cdot \vec{e} \\ &= \vec{e} \end{aligned}$$

Now given $\vec{e} = \vec{m} + \vec{i}$, with $\vec{i} \in I$ it is easy to obtain the original message by reducing $\vec{e}$ modulo $\mathbf{B}_I$.

4.2.2 Evaluation

Much of what is presented in this section comes from (Gentry et al., 2009) and (Armknecht et al., 2015). In coming across all these new definitions and notations the reader would be forgiven if he finds it a bit overwhelming at first. What might these abstract definitions easier to grasp is the observation that the evaluation algorithm uses two different circuits. To begin, it uses a $\pmod{\mathbf{B}_I}$ circuit C (see section 2.4.5) to the plaintexts. Next, it uses a circuit related to C to the ciphertexts which uses ring operations - not $\pmod{I}$. This is what we call a generalised circuit.

Definition 4.2. Let C be a $\bmod \mathbf{B}_I$ circuit. We call generalised circuit $g(C)$ of C the circuit formed by replacing C's $\mathbf{Add}_{\mathbf{B}_J}$ and $\mathbf{Mult}_{\mathbf{B}_I}$ operations with addition and multiplication in the ring R.

Definition 4.3. Let

$$\mathcal{C}_{\mathcal{E}'} = \{C : \forall(x_1, \ldots, x_t) \in X_{\text{Enc}}^t, g(C)(x_1, \ldots, x_t) \in X_{\text{Dec}}\} \tag{4.2.1}$$

This is to say $\mathcal{C}_{\mathcal{E}'}$ is the set of mod-$\mathbf{B}_I$ circuits that, when generalised, its output is always in $X_{\mathbf{Dec}}$ if the inputs are in X_{Enc}. We say $\mathcal{C}_{\mathcal{E}}$is a set of **permitted circuits** if $\mathcal{C}_{\mathcal{E}} \subseteq \mathcal{C}_{\mathcal{E}'}$.

Definition 4.4. We say ψ is a **valid ciphertext** with respect to $\mathcal{E}$, public key pk, and permitted circuits $\mathcal{C}_{\mathcal{E}}$ if it equals **Evaluate**(pk, C, Ψ) for some $C \in \mathcal{C}_{\mathcal{E}}$, where each $\psi \in \Psi$ is the image of **Encrypt**.

Theorem 4.5. *Assume $C_{\mathcal{E}}$ is the set of permitted circuits containing the identity circuit.Then $\mathcal{E}$ is correct for $C_{\mathcal{E}}$.*

Remark. This theorem tells us that **Decrypt** correctly decrypts valid ciphertexts.

Proof. For $\Psi = \{\psi_1, \dots, \psi_t\}, \psi_k = \pi_k + i_k + j_k$, where $\pi_k \in \mathcal{P}, i_k \in I, j_k \in J$ and $\pi_k + i_k \in X_{\text{Enc}}$, we have:

$$\begin{aligned} Evaluate(pk, C, \Psi) &= g(C)(\Psi) \pmod{\mathbf{B}_J^{pk}} \\ &\in g(C)(\pi_1 + i_1, \dots, \pi_t + i_t) + J \end{aligned}$$

Now, if $C \in \mathcal{C}_{\mathcal{E}}$, then by definition 1.2 we have $g(C)(X_{\text{Enc}}, \dots, X_{\text{Enc}}) \in X_{\text{Dec}}$, which means that:

$$\begin{aligned} \text{Decrypt}(sk, \text{Evaluate}(pk, C, \Psi)) &= g(C)(\pi_1 + i_1, \dots, \pi_t + i_t) \pmod{\mathbf{B}_I} \\ &= g(C)(\pi_1, \dots, \pi_t) \pmod{\mathbf{B}_I} \\ &= C(\pi_1, \dots, \pi_t) \end{aligned}$$

which proves the result. □

The key takeaway from this theorem is that the somewhat homomorphically encryption scheme that Gentry introduced is correct for permitted circuits. That is to say it is fully homomorphic for *some* set. The idea now is to extend or maximise this set of permitted circuits.

Definition 4.6. Let r_{Enc} be the smallest value such that $X_{\text{Enc}} \subseteq \mathcal{B}(r_{\text{Enc}})$, where $\mathcal{B}(r)$is the ball with radius r. Let r_{Dec} be the largest such that $\mathcal{B}(r_{\text{Dec}}) \subseteq X_{\text{Dec}}$

Thanks to this definition we may now define permitted circuits as:

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

This new interpretation transform the problem of maximising the set of permitted circuits into a geometrical problem. To maximise the the set $\mathcal{C}_{\mathcal{E}}$ we need to bound the Euclidean norm $||g(C)(\vec{e}_1, \dots, \vec{e}_t)||$.

Theorem 4.7. *For a generalised circuit $g(C)$ of permitted circuit C and lattice vectors $\vec{e}_i, i = 1, \dots, t$ the Euclidean norm of $||g(C)(\vec{e}_1, \dots, \vec{e}_t)||$ can be bounded in terms of $||\vec{e}_i||$ and $||\vec{e}_j||$ such that:*

$$||\vec{e}_i + \vec{e}_j|| \leq ||\vec{e}_i|| + ||\vec{e}_j|| \tag{4.2.2}$$

$$||\vec{e}_i \times \vec{e}_j|| \leq \gamma(R) \cdot ||\vec{e}_i|| \cdot ||\vec{e}_j|| \tag{4.2.3}$$

Proof. The first inequality follows from the the triangular inequality. We have that $\times$ in this situation denotes the polynomial multiplication

$$a(x) \times b(x) = \sum_{i=0}^{n} \sum_{j=0}^{i} a_j b_{i-j} x^{i+j}$$

where $n = deg(a(x) + b(x))$. This is a bilinear map and its operator norm can be defined as:

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

□

In his (Gentry, 2009, Theorem 7.3.2, p. 71), Gentry applies this result to show that the somewhat homomorphic encryption scheme that we just have outlined can correctly evaluate circuits of depth up to:

$$\log \log r_{Dec} - \log \log (\gamma(R) \cdot r_{Enc}) \tag{4.2.4}$$

This tells us then that to maximise the depth of circuits that one may correctly evaluate, we need to minimise the expansion factor $\gamma(R)$ and r_{Enc} while maximising r_{Dec}.

Homomorphic Properties

1. **Addition** Given two plaintext vectors $\vec{m}_1, \vec{m}_2 \in \{0,1\}^n$, we may verify the additive homomorphic property as follows:

$$\begin{aligned} \vec{c}_1 + \vec{c}_2 &= E(\vec{m}_1) + E(\vec{m}_2) \\ &= \left(\vec{m}_1 + \vec{r}_1 \cdot B_i + \vec{g}_1 \cdot b_J^{pk}\right) + \left(\vec{m}_2 + \vec{r}_2 \cdot B_i + \vec{g}_2 \cdot b_J^{pk}\right) \\ &= \vec{m}_1 + \vec{m}_2 + (\vec{r}_1 + \vec{r}_2) \cdot B_I + (\vec{g}_1 + \vec{g}_2) \cdot B_J^{pk} \end{aligned}$$

   From our calculations we see that $\vec{c}_1 + \vec{c}_2$ preserves the format and is within the ciphertext space. To decrypt the sum of the ciphertext one computes $(\vec{c}_1 + \vec{c}_2) \pmod{b_J^{pk}}$ which equals $\vec{m}_1 + \vec{m}_2 + (\vec{r}_1 + \vec{r}_2) \cdot B_I$ for the ciphertexts whose noise amount is smaller than $\frac{1}{2} B_J^{pk}$. For larger

2. **Multiplication** We may similarly examine the homomorphic multiplicative property of Gentry's encryption by examining the product of two ciphertext closely. We have:

$$\begin{aligned} \vec{c}_1 \times \vec{c}_2 &= E(\vec{m}_1) \times E(\vec{m}_2) \\ &= \left(\vec{m}_1 + \vec{r}_1 \cdot B_i + \vec{g}_1 \cdot b_J^{pk}\right) \times \left(\vec{m}_2 + \vec{r}_2 \cdot B_i + \vec{g}_2 \cdot b_J^{pk}\right) \\ &= \vec{m}_1 \times \vec{m}_2 + (\vec{m}_1 \times \vec{r}_2 + \vec{m}_2 \times \vec{r}_1 + \vec{r}_1 \times \vec{r}_2) \cdot B_I \end{aligned}$$

   If the noise $|\vec{e}_1 \times \vec{e}_2|$ is sufficiently small enough the multiplication of plaintexts $\vec{m}_1 \times \vec{m}_2$ can be correctly recovered from the multiplication of ciphertexts $\vec{c}_1 \times \vec{c}_2$.

From the discussion just had we recall that homomorphic operations can be applied only to ciphertexts when the ciphertexts have a small amount of noises.That is to say if the noise parameter $\vec{m} + \vec{r} \cdot B_I$ is close to a lattice point then we may proceed with further addition and multiplication operations. Nonetheless, at some point, we will find that it no longer is possible to decrypt the ciphertext properly, This means that scheme above remains, for now, a somewhat homomorphic encryption scheme as the number of operations are limited. Still it is a significant improvement on BGN, which has a small plaintext space - $\log L$ bits for security parameter L. Gentry's SWHE allows not only greater multiplicative depth - while essentially allowing arbitrary number of addition operations - but also a larger plaintext space.

Since the noise grows faster for with the multiplication operations, we have that the number of possible multiplication operations is less than that of addition.

What to do then? Here is where Gentry introduced his bootstrapping technique. This is essentially a recrypting procedure that produces a so-called fresh ciphertext from the noisy ciphertext corresponding to the same plaintext.

4.3 Squashing

The key to understand what the squashing transformation does is this: it places a hint about the secret key $\vec{v}_{sk}$ inside the public key pk.

One way of trying to understand this better is to think of it through an analogy. Let us suppose that in a Alice's jewellery store, a chemical reaction is used as a key to open a glove box. For an employee to open a box, they must use gloves to rub the key against the inner box until the box dissolves. It turns out, however, that the reaction is too slow and the gloves become stiff before the box dissolves. How can this be solved? Alice may provide each employee accelerants, each box having its own version, that the employee can put on the outside of box $\#i$ just before placing it inside box $\#(i+1)$. The effect is that the chemical reaction between the key and the box is heightened, enabling the reaction to terminate before the gloves grow stiff. The accelerant contains some information about the chemical component of the key, but on its own, it remains insufficient to be used by an employee to construct a key.

4.3.1 SplitKey

This is where a hint about the secret key is placed inside the public key. What does this hint look like? It consists of a random set of vectors $r = \{\vec{t_1}, \ldots, \vec{t_r}\} \in J^{-1}$ and a secret subset of vectors which add up to the original private key:

$$\vec{v}_{sk*} = \sum_{i \in S} \vec{t_i} \pmod{I}$$

In this context S represents the distinguished subset of indices $S \subseteq \{1, \ldots, r\}$ for which the equation above is valid. The input it takes consists of the public and private key as produced by the original encryption procedure and outputs a tuple (sk, r). The new private key is now a matrix - $sk = \mathbf{SK}$ - such that:

$$sk_{ij} = \begin{cases} 1 & \text{if and only if } j \text{ is the } i\text{th member of } S \\ 0 & \text{else} \end{cases}$$

$r$is then added to the original public key pk to yield a new public key $pk*$.

4.3.2 ExpandCT

Evaluation

The next procedure, poignantly called ExpandCT, serves to ensure that the ciphertext is able to handle a new shallower decryption circuit. This is completed by the encrypter, rather than the decrypter, so as to reduce the computational complexity of decryption. ExpandCT computes:

$$\vec{x}_i = \vec{t}_i \times \vec{c} \pmod{B_I}$$

where $i = 1, \dots, r$ and $\vec{c}$ is the ciphertext output produced by Enc. The new ciphertext is the tuple $\psi = \{\vec{c}, \vec{x}_1, \dots, \vec{x}_r\}$.

Decryption

By taking the new privat key **SK** as well as the new ciphertext ψ as input, the decryption algorithm *Dec* allows the user to extract the message using:

$$\vec{m} = \vec{c} - \lfloor \sum_{i \in S} \vec{c}_i \rceil \pmod{B_I}$$

Proof. In order to extract the relevant $\vec{x}_i$'s, the decryptor computes a set of vectors $\{\vec{w}_{ij}\}$with $i = 1, \dots, s$ and $j = 1, \dots, r$ with:

$$\vec{w}_{ij} = sk_{ij} \cdot \vec{c}_j$$

We have that a vector $\vec{w}_{ij} \neq 0$ if and only if $sk_{ij} \neq 0$. This ensures that one obtains the desired $\vec{x}_i$'s, which are those where $i \in S$. By recalling the original decryption algorithm (cite eq?) we have that:

$$\begin{aligned}
\vec{m} &= \vec{c} - \lfloor \vec{c}_{sk} \times \vec{c} \rceil \pmod{B_I} \\
&= \vec{c} - \lfloor \left(\sum_{i \in S} \vec{t}_i \right) \times \vec{c} \rceil \pmod{B_I} \\
&= \vec{c} - \lfloor \sum_{i \in S} \vec{t}_i \times \vec{c} \rceil \pmod{B_i} \\
&= \vec{c} - \lfloor \sum_{i \in S} \vec{x}_i \rceil \pmod{B_I}
\end{aligned}$$

□

What Gentry did is that he then went on to show that this scheme is able to evaluate its decryption circuit, thus making it bootstrappable (Gentry, 2009, chap 10.3).

As part of this transformation we have that the security assumptions come to change a bit. By introducing $r$in the public key, Gentry require another difficult computational problem to ensure secure communication. This is called the **sparse vector subset sum problem** (SVSSP)

Definition 4.8. Let S and T be two natural numbers such that $S \ll T$ and let q be a prime number. The challenger sets $b \leftarrow \{0, 1\}$. If $b = 0$, it generates a set τ whose cardinality is $|\tau| = T$ of uniformly random integers in $[-\frac{q}{2}, \frac{q}{2}]$ such that there exists a subset of cardinality S whose elements sum to $0 \pmod{q}$. The problem to be solved is to guess b.

4.3.3 Bootstrapping

The notion of recryption which works by encrypting a ciphertext anew (so that it becomes doubly encrypted) and then removing the inner encryption by homomorphically evaluating the doubly encrypted plaintext and the encrypted decryption key using the decryption circuit. As long as the evaluation algorithm can handle the decryption process plus one more gate, progress can be made in evaluating the circuit of interest.

Definition 4.9. Let $C_{\mathcal{E}}$be a set of circuits with respect to which $\mathcal{E}$ is homomorphic. $\mathcal{E}$ is said to be **bootstrappable** with respect to Γ if:

$$D_{\mathcal{E}}(\Gamma) \subseteq C_{\mathcal{E}}$$

How is this done? Firstly, we generate two different public and secret key pairs (pk_1, sk_1) and (pk_2, sk_2). The private keys are kept by the client (Alice), while the public keys are shared with the server (Bob). Next, the encryption of the private key, $E_{pk1}(sk_1)$ is also sent to the server (Bob), who at this point already possess $c = E_{pk_1}(m)$. As the scheme presented above already can evaluate its own decryption algorithm homomorphically, the noisy ciphertext is decrypted homomophically using $E_{pk1}(sk_1)$. This result is then encrypted using a different public key, namely pk_2, such that $E_{pk_2}(D_{sk_1}(c)) = E_{pk_2}(m)$.

To put it even more succinctly: The first decryption of the noisy ciphertext removes the noise, while the new homomorphic encryption produces a small(er) noise to the ciphertext. This permits us to compute more homomorphic operations on this ”fresh” ciphertext until we reach (again) the threshold point. We may now describe Gentry's construction of a fully homomorphic encryption scheme as follows: Begin by constructing a somewhat homomorphic encryption scheme and then apply the squashing method to reduce the circuit depth of the decryption algorithm. Then apply the bootstrapping technique to obtain a fresh ciphertext. Since bootstrapping can be applied repeatedly, we are now in a position to compute unlimited number of operations on the ciphertexts, which is to say we have successfully constructed a fully homomorphic encryption scheme.

It is worth noticing that Gentry's bootstrapping solution increases the computational cost significantly, and this is indeed one major drawback for practically implementing it.

Again, it may be instructive to think of bootstrapping using an analogy (Gentry, 2009, p.10). Let us return to the Alice's jewellery store. You will recall that Alice out of paranoia for having any of her employees steal her jewellery placed them in special boxes that employees could work on using particular gloves. Now, imagine that Alices glove boxes are defective; meaning the gloves stiffen after after an employee uses the gloves for 1 minute. None of Alice's employees are able to put together her designs in that short amount of time. But Alice has a solution. She gives to an employee that is putting together her designs a glove box containing the raw materials, but also several additional glove boxes. Each of these additional glove boxes holds a copy of her master key. To assemble the intricate design, the employee manipulates the materials in box #1 until the gloves stiffen. Next, he places box #1 inside box #2, where the latter box already contains a master key. Using the gloves for box #2, he opens box #1 with the master key, extracts the partially assembled trinket, and continues the assembly within box #2 until its gloves stiffen. He then places box #2 inside box #3, and so on. When the employee finally finishes his assembly inside box #n, he hands the box to Alice. It is important to note that this trick will not work unless the employee can open box #i within box #$(i + 1)$, and have time to make a little bit of progress on the assembly, all before the gloves of box #$(i + 1)$ stiffen.

What we see here is analogous with the requirement for a bootstrappable encryption scheme $\mathcal{E}$, namely the augmented decryption circuit complexity of $\mathcal{E}$ is less than what $\mathcal{E}$ is able to homomorphically evaluate.

5 Applications

Now that we have had the opportunity to familiarise ourselves with the background, theory, and underlying security assumptions of some homomorphic encryption schemes it may be of interest to learn more about possible applications. We should note that even in instances when there are other methods available FHE-solutions often remain more conceptionally appealing as they are simpler and easier to explain. In what follows we list a couple of examples that demonstrate the underlying potential of FHE when applied to practical problems.

5.1 Outside storage

Consider a situation where a financial institution possess both sensitive data and also proprietary algorithms that they wish to keep secret. It could be an algorithm that based on the data acquired predicts stocks or bond values. (Naehrig, Lauter, & Vaikuntanathan, 2011) brought forward the proposal to apply homomorphic encryption to upload both the data and the algorithm in encrypted form so as to allow outsourcing of the computations to a cloud service.

What is worth pointing out though is that the process of ensuring the algorithm remains secret is not due to homomorphic encryption. Instead it is part of **obfuscation** research. The closest solution to this that homomorphic encryption offers is known as **circuit privacy**. It, however, only guarantees that information about the function does not leak by the output - not that one can encrypt the function itself.

The problem homomorphic encryption offers to solve is rather as follows: Suppose Alice possess sensitive data, such as a portfolio of stocks, and Bob has proprietary algorithms that make predictions about stock prices. In case Alice would want to apply Bob's algorithm, either she would have to divulge her stock portfolio to Bob, or Bob would have to give the algorithm to Alice. With homomorphic encryption, however, Alice can encrypt the data with a circuit private scheme and send it to Bob, who runs the proprietary algorithm and only sends back the result, which can only be decrypted by Alices private key. Accordingly Bob cannot learn anything about Alice's data, and Alice will not have any access to the algorithm's Bob used.

5.2 Consumer Privacy

As much as most of us dislike advertisement there are instances when targeted, specifically tailored advertisement is useful to serve the needs of customers. Still, many users express concern about the privacy of their data, which for our example could be the particular preference of books we are looking at or where we are currently staying.

(Armknecht & Strufe, 2011) introduced a recommender system where a user receives encrypted recommendations without the the system being aware of the content. At the centre a rather straightforward, but highly efficient homomorphic encryption scheme, which has been developed exclusively for this purpose. As such they are able to have a function to be computed that chooses the advertisement for each user while the advertising remains encrypted.

(Jeckmans, Peter, & Hartel, 2013) use the somewhat homomorphic encryption scheme of (Brakerski & Vaikuntanathan, 2011). As such, users are able to obtain recommendations from friends without the identities of the recommenders being revealed. The one requirement though is that the advertisements are form a third party and that there is no collusion with the provider.

5.3 Medical Applications

One of the most promising areas of research right now is bioinformatics. With the onslaught of new technologies and as researchers develop new methods for analysing genomic data the cost **and** effectiveness of human genome sequences will only improve. As analysing genomic data improves the cost of sequencing human genome drops thus increasing the quantity of data available for scientific examination. Nevertheless, to enable researchers to access their data, patients expose themselves to risks from invasion of privacy (Humber, 2013). As (Ayday, Cristofaro, Hubaux, & Tsudik, 2013) have remarked, even anonymised data can be re-identified on an individual basis "using information available from popular genealogy web sites and other available information".

There are multiple approaches to protect privacy while ensuring the continuation of research on data. These include policy-based solutions, de-identification of data, approximate query answering, as well as technological solutions based on cryptography. The last point is where homomorphic encryption fits in.

(Lauter & Naehrig, 2014) propose using a homomorphic encryption scheme to address this problem. In particular, they show how to encode genotype and phenotype data for encryption and how to apply the Pearson Goodness-of-Fit test, the D' and r^2-measures of linkage disequilibrium, the Estimation Maximisation (EM) algorithm for haplotyping, and the Cochran-Armitage Test for Trend. Each of these are standard algorithms that are widely applied in genome analyses. In this scenario, the user is the data owner, which means that the encrypted data is under the user's public key and only he/she can decrypt. With the data encrypted, the service allows doctor to access vital information such as heart rate, blood pressure, weight etc. to predict the chances of certain conditions occur.

5.4 Private Queries

Another straightforward application is to enable private queries to a database or a search engine. **Private information retrieval** is arguably the simplest example to illustrate this application (Chor, Goldreich, & Kushilevitz, 1998). Here, there is a server holding a large database (i.e. the Swedish Land Registry database), and a client who wants to retrieve one record of this database without the server learning which record was retrieved. Homomorphic encryption makes this happen by encrypting the index of the record the person wishes to retrieve. The server evaluates the function $f_{db}(i) = db[i]$ on the encrypted index, before returning the encrypted result to the client who then can decrypt it and obtain the plaintext record.

6 Limitations

Despite earning the nickname "the Swiss Army knife of cryptography" (Barak, 2012) for its versatility, homomorphic encryption does not resolve all issues in cryptography. In this section, we will briefly focus on three aspects where homomorphic encryption in some manner fall short.

6.1 Inputs are all encrypted by the same key

Homomorphic encryption only processes encrypted data with one single key. There are instances though when one may want to be able to process data that used several keys for encryption. Case in point, imagine a team of researchers, each with its key, uploading their encrypted data to the cloud. At this stage, we may want the cloud to aggregate this data and compute useful statistics on it. Naturally, one would only be able to recover the plaintext result only if all the parties joined to cooperate, bringing its corresponding private key. A homomorphic encryption scheme that manages this is referred to as a **multikey homomorphic**.

Multikey homomorphic encryption is a new concept introduced by (López-Alt, Tromer, & Vaikuntanathan, 2012). While t also is not without any shortfalls. For instance one has to know the upper bound on the number of parties at the time of generating the keys, as the parameters grow with the number of parties involved.

(Clear & McGoldrick, 2015) recently proposed a different realisation under LWE, that would later be significantly simplified by (Mukherjee & Wichs, 2016).

6.2 The output is encrypted

While homomorphic encryption allows us to apply arbitrary functions on the encrypted data, the outcome of such computation is itself a ciphertext, which one can only make sense of with the private key. There are, however, many instances when one would like to process encrypted data and get (only) the result of the computation in the clear. Let us consider two different examples.

Firstly, let us consider a situation where we want to apply a spam filter to an encrypted mail. While we wish to maintain the content of the e-mail secret we may also want the server to learn to distinguish between spam/no-spam mails, thus enabling it to only forward non-spam messages.

Secondly, imagine that we have produced a new model for predicting the risk of cancer based on a number of factors. We now wish to wish to release this to the general public. But, we also want to withhold the inner mechanics of the model, either for concern over intellectual property, or because we need to preserve the privacy of patient data that we used to devise the model. In other words, we want the model itself to be encrypted, but anyone should be able to evaluate the model on their own indicators and get a decisive result.

What these two examples illustrate are typical applications of **functional encryption** (the first example) and **code obfuscation** (the second example).

6.3 No guarantees of integrity

While homomorphic encryption permits us to compute on encrypted data, it does not provide any guarantees of integrity for the computed values. That is to say that if we were to encrypt an input message m using the encryption function e we have no guarantees that the produced ciphertext c was produced by evaluating e on m.

Verifiable computation, the process of verifying the integrity of remote computation, addresses this issue. As one might expect, this is a very active research area at the moment as (Walfish & Blumberg, 2015) outlines in a survey of the practical implications of verifiable computation.

7 Summary

This year marks the 40th anniversary of the publication of the RSA-encryption scheme, which in many ways represents the beginning of the long journey towards realising a fully homomorphic encryption scheme. Back when it was first introduced, the notion of being able to operate on encrypted data would have seemed of little relevance for the common person. Fast forward to today, in the digital/information age we find ourselves living in, and one cannot help but to think that privacy of data is a more pressing issue than at any time before. Some of our time's most dominant industries, including e-banking and online retail, find it crucial to protect their users' assets and accounts from malicious third-parties. The current paradigm lets the user encrypt his data and share the keys with the service provider. That is to say the individual is not in control of his sensitive data.

This is where Homomorphic Encryption schemes enter as a promising direction to take going forward. They allow any third party to operate on the encrypted data without decrypting it in advance. Even though the idea of a fully homomorphic encryption scheme had been around for more than 30 years, it was not until Craig Gentry introduced his paradigm-shifting scheme in 2009 that it was finally realised theoretically. Since then, others have contributed either development of this particular scheme or by adding other approaches to achieve fully homomorphic schemes. In the process, they have demonstrated that even though much progress has been made, there is still much that needs to be improved until fully homomorphic encryption schemes can be employed in practical applications in real-life.

In this brief overview, we have covered some of the critical milestones leading up to the realisation of fully homomorphic encryption scheme, including presentations of partially homomorphic schemes somewhat homomorphic schemes, and lattice theory, as well as providing a more gentle introduction of Gentry's groundbreaking result. Moreover, we have also had the opportunity to touch upon various aspects of applications.

Going forward our focus should be on making these theoretical breakthroughs accessible to practical applications, such as those outlined in the final section, by developing more efficient encryption schemes.

References

Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C. A., & Strand, M. (2015). A guide to fully homomorphic encryption. *IACR Cryptology ePrint Archive*, *2015*, 1192.

Armknecht, F., & Strufe, T. (2011). An efficient distributed privacy-preserving recommendation system. In *Ad hoc networking workshop (med-hoc-net), 2011 the 10th ifip annual mediterranean* (pp. 65–70).

Ayday, E., Cristofaro, E. D., Hubaux, J., & Tsudik, G. (2013). The chills and thrills of whole genome sequencing. *CoRR*, *abs/1306.1264*. Retrieved from `http://arxiv.org/abs/1306.1264`

Barak, B. (2012). *The swiss army of cryptography.* Retrieved 2017-08-23, from `https://windowsontheory.org/2012/05/01/the-swiss-army-knife-of-cryptography/`

Beachy, J. A., & Blair, W. D. (2006). *Abstract algebra.* Waveland Press.

Bellare, M., Boldyreva, A., & ONeill, A. (2007). Deterministic and efficiently searchable encryption. *Advances in Cryptology-CRYPTO 2007*, 535–552.

Benaloh, J. (1994). Dense probabilistic encryption. In *Proceedings of the workshop on selected areas of cryptography* (pp. 120–128).

Boneh, D., Goh, E.-J., & Nissim, K. (2005). Evaluating 2-dnf formulas on ciphertexts. In *Tcc* (Vol. 3378, pp. 325–341).

Brakerski, Z., & Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference* (pp. 505–524).

Chor, B.-Z., Goldreich, O., & Kushilevitz, E. (1998, December 29). *Private information retrieval.* Google Patents. (US Patent 5,855,018)

Clear, M., & McGoldrick, C. (2015). Multi-identity and multi-key leveled fhe from learning with errors. In *Annual cryptology conference* (pp. 630–656).

Diffie, W., & Hellman, M. (1976, Nov). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644-654. doi: 10.1109/TIT.1976.1055638

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, *31*(4), 469–472.

Gentry, C. (2009). *A fully homomorphic encryption scheme.* Stanford University.

Gentry, C., et al. (2009). Fully homomorphic encryption using ideal lattices. In *Stoc* (Vol. 9, pp. 169–178).

Goldwasser, S., & Bellare, M. (1996). Lecture notes on cryptography. *Summer course Cryptography and computer security at MIT*, *1999*, 1999.

Goldwasser, S., & Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual acm symposium on theory of computing* (pp. 365–377).

Goldwasser, S., & Micali, S. (1984). Probabilistic encryption. *Journal of computer and system sciences*, *28*(2), 270–299.

Hoffstein, J., Pipher, J. C., Silverman, J. H., & Silverman, J. H. (2014). *An introduction to mathematical cryptography* (Vol. 2). Springer.

Humber, J. M. (2013). *Biomedical ethics and the law.* Springer Science & Business Media.

Jeckmans, A., Peter, A., & Hartel, P. (2013). Efficient privacy-enhanced familiarity-based recommender system. In *European symposium on research in computer security* (pp. 400–417).

Katz, J., & Lindell, Y. (2014). *Introduction to modern cryptography.* CRC press.

Lauter, K., & Naehrig, M. (2014, June). Private computation on encrypted genomic data.

López-Alt, A., Tromer, E., & Vaikuntanathan, V. (2012). On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual acm symposium on theory of computing* (pp. 1219–1234).

Mukherjee, P., & Wichs, D. (2016). Two round multiparty computation via multi-key fhe. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 735–763).

Naehrig, M., Lauter, K., & Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd acm workshop on cloud computing security workshop* (pp. 113–124).

Paillier, P., et al. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt* (Vol. 99, pp. 223–238).

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126.

Walfish, M., & Blumberg, A. J. (2015). Verifying computations without reexecuting them. *Communications of the ACM*, *58*(2), 74–84.