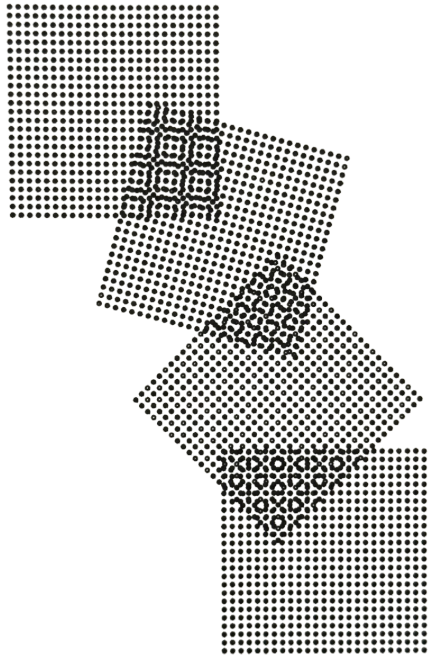
Homomorphic encryption: new constructions and algorithms

Lorenzo Rovida



Homomorphic encryption: new constructions and algorithms

Disclaimer
Note on the titlepage: the cover art represents a Moiré pattern, evoking a lattice and its rotations, drawn by Adam Fuhrer.

Colophon
This document was typeset with the help of KOMA-Script and LaTeX using the kaobook class. The source code for the template of this book is available at:
https://github.com/fmarotta/kaobook

Università degli Studi di Milano - Bicocca
Faculty of Science
Department of Informatics, Systems and Communication
Ph.D. program in Computer Science, XXXVIII cycle

Homomorphic encryption: new constructions and algorithms

By:

Lorenzo Rovida

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy in Computer Science

October 2025

Supervisor: *Prof. Alberto Leporati*

Tutor: *Prof. Michele Ciavotta*

Coordinator: *Prof. Leonardo Mariani*

Abstract

The main focus of this thesis is the *Fully homomorphic encryption* (FHE) primitive. This is a cryptographic technique that allows one to encrypt values in such a way that a party that does not own the secret key can still perform computations over encrypted values. To have an intuition, think the following (homomorphic) property:

$$a + b = \text{dec}(\text{enc}(a) + \text{enc}(b)),$$

where $\text{enc}(\cdot)$ and $\text{dec}(\cdot)$ are the encryption and decryption functions, respectively. An external (untrusted) party could evaluate $\text{enc}(a) + \text{enc}(b)$, and by returning the result back to the user, the latter will be able to decrypt it and to see result of the computation, even though she never published neither a or b.

By extending this definition to additions and multiplications, and by allowing for the evaluation of arbitrary depth circuits, we precisely obtain FHE. The range of potential use cases of this technique is enormous: from privacy-preserving machine learning to private information retrieval, although it inherently introduces some computational overhead that prevents current solutions to be deployed as real world products – the goal of this work is to speed up this process. After an initial introduction to *lattice-based cryptography*, the thesis is split into two parts: *constructions* and *algorithms*.

In the first part we will begin by presenting how such schemes are built, starting from existing ones, and we will propose a generalization for these techniques. In particular, all standard FHE schemes are based on the hardness of the *Learning with errors* (LWE) problem. This problem can be seen from multiple perspectives, and we will take the lattice one. From a certain perspective, an LWE instance can be seen as a *Bounded distance decoding* (BDD) problem over a specific family of lattices, called q-ary lattices. Our objective is to generalize the lattice properties required to build an FHE scheme, enabling the use of other lattice assumptions. We will show how to do so, and we will give as an example a possible FHE scheme based on the *Lattice isomorphism problem*, that enables to sample BDD problems over any (meaningful) family of lattices, by hiding the BDD target in a (secret) rotation of the chosen lattice.

In the second part of this thesis we will construct algorithms that allow one to efficiently compute some function $f(x)$, given an encrypted input x. We will present different contributions, mainly in the field of *privacy-preserving machine learning*. These algorithms allow one to classify encrypted data in such a way that the data owner *only* will be able to read the result of the classification. As a potential use case, think of search engines that are able to return the results of some query, without being able to see what the user asked. Unfortunately, because of the overhead introduced by FHE, we need to take a step back and work on simpler problems.

In particular, we will show how to efficiently classify images from the CIFAR-10 dataset with a drastically smaller memory footprint, but with the same performance as the state of the art. Some of the main issues are about how to perform an efficient convolution with encrypted data, and how non polynomial functions – such as $\text{sgn}(x)$ – can be

evaluated, having only access to additions and multiplications. Additionally, given the current interest in *Large language models*, we will give some preliminary exploration of the Transformer architecture under FHE, allowing us to classify encrypted text from the SST-2 dataset.

Lastly, we will deal with sorting algorithms for encrypted data. Standard results can not be applied in our domain, since most of the sorting algorithms are data-dependent – meaning that their behavior depends on the input data. Since, in our setting, inputs are encrypted, we can only evaluate oblivious algorithms. We will show how sorting for encrypted values can be made faster, with larger precision and with smaller memory requirements than the state-of-the-art.

Riassunto

Il tema principale di questa tesi è la primitiva crittografica chiamata *Fully Homomorphic Encryption* (FHE) (in italiano, *cifratura omomorfa*). Si tratta di una tecnica crittografica che consente di cifrare dei valori in modo tale che un'entità esterna, priva della chiave segreta, possa comunque eseguire operazioni sui dati cifrati. Per avere un'intuizione di base, si consideri la seguente proprietà (omomorfa):

$$a + b = \mathrm{dec}(\mathrm{enc}(a) + \mathrm{enc}(b)),$$

dove $\mathrm{enc}(\cdot)$ e $\mathrm{dec}(\cdot)$ sono le funzioni di cifratura e decifratura, rispettivamente. Un soggetto esterno (non fidato) può calcolare $\mathrm{enc}(a) + \mathrm{enc}(b)$ e restituire il risultato all'utente, che potrà così ottenere l'esito della computazione pur non avendo mai rivelato né a né b.

Estendendo questa definizione alle operazioni di somma e moltiplicazione, e permettendo la valutazione di qualsiasi algoritmo, si ottiene il concetto di FHE. Le potenziali applicazioni di questa tecnica sono numerosissime; tuttavia, essa introduce un pesante sovraccarico computazionale che, ad oggi, limita l'utilizzo pratico di FHE nel mondo reale. Lo scopo di questo lavoro è contribuire ad accelerare il percorso verso implementazioni realmente utilizzabili. In seguito ad una parte introduttiva alla crittografia basata sui reticoli (*lattice-based cryptography*), la tesi è suddivisa in due parti: costruzioni e algoritmi.

Nella prima parte, presenteremo le modalità con cui tali schemi vengono costruiti, partendo da quelli esistenti, e proporremo una generalizzazione di queste tecniche. In particolare, tutti gli schemi FHE standard si basano sulla difficoltà del problema chiamato *Learning with errors* (LWE). Questo problema può essere visualizzato da diverse prospettive; noi seguiremo quella dei reticoli (*lattices*). Più precisamente, un'istanza di LWE può essere vista come un problema di *Bounded distance decoding* (BDD) su una specifica famiglia di reticoli, chiamati q-ary lattices. Il nostro obiettivo è generalizzare le proprietà dei reticoli necessarie alla costruzione di uno schema FHE, in modo da poter utilizzare altre tipologie di *trapdoor* basate su reticoli. Mostreremo una possibile generalizzazione e presenteremo, come esempio, un possibile schema FHE basato sul *Lattice isomorphism problem*, che consente di generare problemi BDD su qualsiasi (significativa) famiglia di reticoli, nascondendo il target BDD in una rotazione segreta del reticolo scelto.

Nella seconda parte, costruiremo algoritmi in grado di calcolare in modo efficiente una funzione $f(x)$, dato un input cifrato x. Presenteremo diversi contributi, principalmente nell'ambito del *privacy-preserving machine learning*. Questi algoritmi consentono di classificare dati cifrati in modo tale che solo il proprietario dei dati possa leggere il risultato della classificazione. Un possibile caso d'uso è quello di motori di ricerca in grado di restituire i risultati di una query senza conoscere il contenuto della richiesta dell'utente. A causa del sovraccarico computazionale introdotto da FHE, è prima necessario concentrarsi su problemi di scala più ridotta.

In particolare, mostreremo come sia possibile classificare in modo efficiente le immagini del dataset CIFAR-10, riducendo drasticamente i requisiti in termini di memoria ma senza sacrificare le prestazioni rispetto allo stato dell'arte. Alcune delle principali sfide riguardano l'esecuzione di convoluzioni efficienti su dati cifrati e la valutazione di funzioni non lineari, potendo disporre soltanto di somme e moltiplicazioni. Inoltre, dato il crescente interesse verso i *large language models*, proporremo una prima esplorazione dell'architettura *Transformer* in ambito FHE, applicata alla classificazione di testi cifrati del *dataset* SST-2.

Infine, affronteremo il tema degli algoritmi di ordinamento per dati cifrati. I risultati classici nel campo dell'informatica teorica non possono essere applicati direttamente nel nostro contesto, poiché il loro comportamento dipende dai dati in ingresso. Nel nostro caso, i dati sono cifrati e quindi possiamo valutare soltanto *oblivious algorithms*, cioè algoritmi indipendenti dal contenuto dei dati. Mostreremo come l'ordinamento di valori cifrati possa essere reso più efficiente, con maggiore precisione e minori requisiti di memoria rispetto alle soluzioni attuali.

Contents

1 Introduction to the thesis

This chapter has the objective of gently presenting the context and the main motivations behind this thesis. In particular, the concepts of *homomorphic encryption* and *lattice-based cryptography* will be introduced from an high-level perspective in order to present the main intuitions required to understand the subsequent work. Some familiarity with cryptography is assumed; if this is not the case, we refer the curious reader to, for example, [KY20].

[KY20]: Kats and Yehuda (2020), Introduction to Modern Cryptography (3rd ed.)

Where does this thesis live? We can position this work inside the larger set of techniques that allow to compute on encrypted data. Historically, cryptography has been a very important tool to protect communications over public channels, and in particular to protect data *in transit* or stored in databases. Think of, as an example, communications over the Internet. Once a message reaches its destination, it is decrypted by the recipient. Nevertheless, the introduction of public-key schemes [Mer78; DH76], in the late 70s, unlocked new perspectives that extended the applications of cryptography to an extremely wider range. Today, we can evaluate functions over ciphertexts without decrypting them, to prove the knowledge of a secret without actually revealing it, and many other fancy applications, thanks to new cryptographic constructions.

[Mer78]: R. C. Merkle (1978), *'Secure communications over insecure channels'* . In: Commun. ACM

[DH76]: Diffie and Hellman (1976), *'New directions in cryptography'* . In: IEEE Transactions on Information Theory

Usage of 'we'. Throughout the thesis, 'we' will be used for different reasons. (i) it helps to create a conversational and inclusive tone: it is not just me writing, but it is also you reading. Therefore, I hope to gently guide the reader by being more conversational than impersonal. (ii) 'we' can refer to (a subset of) the cryptologic community; consider, for instance, a sentence like *we believe that the post-quantum migration should be as fast as possible*. Lastly, (iii) 'we' can be used to refer to the point of view of my co-authors and I.

How to read this thesis. The rest of this document is structured in the following way: in this chapter we will present the motivations and we will position the contributions of this thesis in the broader space of cryptology. In Chapter 2, we will introduce all the mathematical ingredients required to better understand these contributions. The rest of the thesis

is divided in two parts. The first, named *constructions*, is covered in Chapter 3 and 4, where we first introduce existent schemes and we propose new cryptosystems with a purely lattice-based perspective. In the second part, *algorithms*, we discuss different implementations of algorithms that work over encrypted data, from classification of encrypted images (Chapter 4), text (Chapter 5) and sorting procedures (Chapter 6).

Why computations over encrypted data? Given that we already have public-key cryptography, one might argue that communications on Internet are already encrypted, so security is already verified for most cases. This is absolutely true and in fact, even though Internet communications are mostly sent through public channels, for an eavesdropper life is not easy as these communications are usually encrypted. Nevertheless, in most applications data is mostly encrypted *in transit* only.

For instance, if Alice sends a message to Bob (Figure 1), this will eventually be decrypted by Bob itself (Figure 2), and this makes absolutely sense. Let us think about a slightly more complex situation though. Say that Bob offers a Wikipedia-like service, and Alice sends a query to Bob, and the latter has the objective of returning the corresponding webpage, i.e., a very long string encoded in some way. Notice that Bob has to decrypt Alice's query in order to return the corresponding webpage, and this makes again absolute sense.

Figure 1: Alice and her message for Bob

Nevertheless, cryptography give us tools that allow Bob to return the same webpage, but without ever seeing Alice's query in plain text. In this setup, only she has the key to open the result, so Bob is returning something which contains the (encrypted) result of the query, without even knowing its content. In this setting, data in encrypted both *in transit* and *in the process*, so it is never visible to Bob. Although counter intuitive, we have different ways to achieve this result.

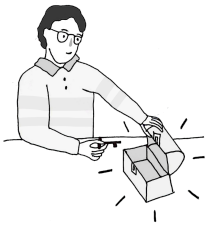


Figure 2: Bob opening Alice's message

In general, there are some well-known techniques in cryptography that allow to evaluate functions over encrypted data, depending on the specific setting. One of the first examples of computations on encrypted data that we can find is due to Yao millionaire's problem[1], which gave rise to *garbled circuits* [Yao86; GMW87], perhaps one of the first techniques that allowed different parties to compute some function over encrypted inputs. Today, we can find different tools that can be used to tackle this problem:

- Trusted execution environment (TEE) [SM24]: there ex-

1: A problem in which two parties with fragile egos owing respectively a and b, want to check whether $a \geq b$ without revealing the values.

[SM24]: Shepherd and Markantonakis (2024), Trusted Execution Environments

ists a family of processors with the very specific property of receiving and returning encrypted data, although computations in the middle are performed on the corresponding decrypted values. Note that this is a very costly hardware solution in which we require to trust hardware vendors and have very limited performance, especially in terms of memory.

- Multi-party computation (MPC) [Lin20]: the core idea is to split some data x between say n parties as $x = x_1 + x_2 + \cdots + x_n$. Note that a party only has access to a portion of x, but they are still able to compute e.g., $x_i + y_i = z_i$. If everyone follows the protocol, by putting together their sharings[2], they will obtain z. In MPC, we have different ways to compute multiplications, unlocking evaluations of arithmetic circuits between many parties, see, e.g., [SM25].
- Fully homomorphic encryption (FHE): the main topic of this thesis, will be extensively discussed. In its standard form (and the one analyzed in this thesis), the setting is composed of a client requesting for a computation and an honest-but-curious[3] server evaluating it. In more advanced situations, we could also define *threshold* FHE [Bon+18], where n parties participate in the protocol and each of them owns a share of the secret key – to decrypt, the parties perform a joint decryption protocol.

[Lin20]: Lindell (2020), *'Secure multiparty computation'* . In: Communications of the ACM

2: A sharing is a way to split a secret value into multiple pieces (shares) and distribute them among parties so that no single party learns the secret, but the secret can be reconstructed when enough shares are combined.

[SM25]: S. Sun and Makri (2025), SoK: Multiparty Computation in the Preprocessing Model (Cryptology ePrint Archive, Paper 2025/060)

3: Who follows the protocol properly and that keeps a record of all its intermediate computations.

These techniques are becoming more and more practical, allowing for real-world applications. One of the goals of this thesis is to speed up this process towards *always* encrypted Internet communications, where data breaches and privacy concerns would not be present anymore. Everything, though, started with a question, firstly posed by Rivest, Adleman and Dertouzos [RAD78]: can we delegate the processing of data without revealing it?

[RAD78]: R. L. Rivest, Adleman, and Dertouzos (1978), *'On data banks and privacy homomorphisms'* . In: Foundations of secure computation

1.1 Homomorphic encryption

Let us start from the intuition behind Homomorphic encryption (HE). Think of a situation in which Alice is putting some values inside a locked chest (Figure 3).

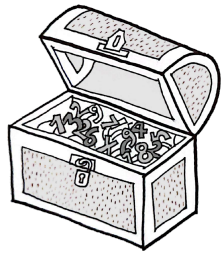


Figure 3: Alice chest containing some numbers

Now, say that Alice closes the chest and only she has access to the key of the padlock. Differently from a standard public-key setting though, this chest is special as it allows for people to "put their hands inside" to operate on its content; think of a

glove box. Alice's chest is sent to Bob, who is able to operate on the encrypted values, or, following the metaphor, to put his hand inside the chest (Figure 4).

Notice that in this protocol, Bob is not able to open the box – as he does not have the secret key – but he is still able to operate in its content. When the latter is returned back to Alice, she will find the result (Figure 5).

Figure 4: Bob operating inside a locked chest

This is the core idea of homomorphic encryption, namely to perform some computation over a set of encrypted values. Intuitively, the encryption ($\text{enc}_{\text{pk}}(x)$) and decryption ($\text{dec}_{\text{sk}}(x)$) operations of a (fully) homomorphic encryption scheme are such that:

- $\text{dec}(\text{enc}(a) + \text{enc}(b)) = a + b$
- $\text{dec}(\text{enc}(a) \cdot \text{enc}(b)) = a \cdot b$

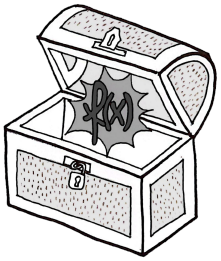


Figure 5: Alice chest after the computation

Before introducing the most powerful *fully homomorphic encryption* (FHE) primitive, let us briefly introduce a weaker version of it, called *partially homomorphic encryption*; starting from the well-known RSA scheme [RSA78].

1.1.1 Partially homomorphic encryption

In 1978, the well-known RSA scheme [RSA78] was introduced. It was the first example of a public-key encryption scheme. Let us briefly remark it, in its most basic form (also known as Textbook RSA, which is not used in practice). The key generator selects two distinct prime numbers p and q and computes their product $n = p \cdot q$. Then, given the Euler's totient function $\phi(n) = (p-1) \cdot (q-1)$, where $n = p \cdot q$, they choose an integer $e \in (1, \phi(n))$ such that $\gcd(e, \phi(n)) = 1$, namely e and $\phi(n)$ are coprime. At this point the secret key is computed as $d \equiv e^{-1} \bmod \phi(n)$, which means that $d \cdot e \equiv 1 \bmod \phi(n)$. Notice that the security relies on the hardness of recovering either $\phi(n)$ or the factorization of n, as this allows in turn to find $\phi(n)$, therefore to easily find d. At this point, the public and the secret key are constructed as $pk := (e, n)$ and $sk := (d, n)$. To encrypt a message $m < n$ we can simply compute

$$c := m^e \bmod n.$$

Then, to retrieve a message, we can use the secret d, for which we know that $d \cdot e \equiv 1 \bmod \phi(n)$, or, in other words, $d \cdot e = 1 + \phi(n) \cdot k$ for some $k \in \mathbb{N}$. We thus compute m as:

$$c^d \equiv (m^e)^d \equiv m \cdot (m^{\phi(n)})^k \equiv m \bmod n.$$

since $m^{\phi}(n) \equiv 1 \bmod n$ by Euler's totient theorem.

We are interested in a very specific property of this version of RSA, namely its *multiplicative homomorphism*, observed in [RAD78]. Given two ciphertexts $c_1 := (m_1)^e \bmod n$ and $c_2 := (m_2)^e \bmod n$, we can indeed compute

$$c^{\times} := c_1 \cdot c_2 \equiv (m_1)^e \cdot (m_2)^e \equiv (m_1 \cdot m_2)^e \bmod n,$$

and by decrypting the newly generated ciphertext $c^{\times}$, one obtains

$$(c^{\times})^d \equiv (m_1 \cdot m_2)^{e \cdot d} \equiv m_1 \cdot m_2 \bmod n,$$

that is precisely the result of the multiplication between the two underlying messages. In other words, we just computed $m_1 \cdot m_2$ from c_1 and c_2, without revealing the two clear values! We give the intuition of this process in Figure 6.

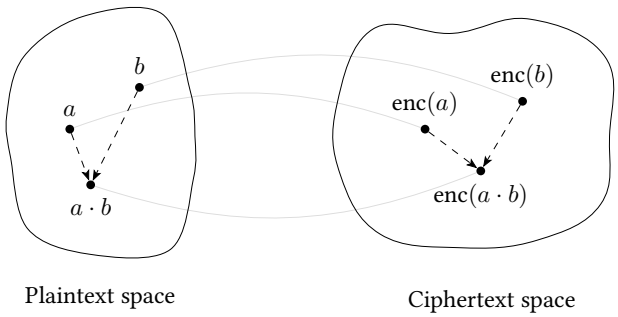


Figure 6: Multiplicative homomorphism between the plaintext and the ciphertext spaces

This is what we call a *partially homomorphic encryption* scheme, namely a scheme that allows to perform one type of operation (either addition or multiplication) between ciphertexts. In literature, we can find other examples of such schemes, e.g. [GM82; ElG85; Pai99].

1.1.2 Somewhat homomorphic encryption

In Textbook RSA, we can not perform an addition between two ciphertexts, as its decryption would not make any sense[4]. Therefore, we now introduce the so-called *somewhat homomorphic encryption* (SWHE) schemes, that allows to perform both additions and multiplications, but with a tight limit on the number of total operations. One of the first examples of such schemes is by Boneh, Goh and Nissim [BGN05], which allows one to evaluate an arbitrary number of additions, followed by

4: Observe that $(m_1^e + m_2^e)^d \not\equiv m_1^{e \cdot d} + m_2^{e \cdot d}$

a single multiplication. After that, additions are allowed but no more multiplications can be carried out.

A trace of luck has surfaced from *noisy* encryption schemes. A very nice example to "softly" introduce the role of *noise* in computations is the DGHV scheme, by van Dijk, Gentry, Halevi and Vaikunthanatan [vDGHV10]. For simplicity, we describe a secret-key version of the scheme, although it can be turned into a public key one with a clever trick introduced by Rothblum [Rot11].

[vDGHV10]: van Dijk, Gentry, Halevi, and Vaikuntanathan, *'Fully Homomorphic Encryption over the Integers'* . In: EUROCRYPT '10

The scheme is based on the hardness of solving the approximate greatest common divisor (AGCD) problem[5], and it is defined as follows. The secret key is an odd integer $p \in [2^{\eta-1}, 2^{\eta})$ for some large $\eta \in \mathbb{N}$ that represents the bit-length of the secret. The (secret key) encryption is performed as

$$c := m + 2r + pq,$$

where $m \in \{0, 1\}$ is the message, $r \ll p \in \mathbb{Z}$ is a sufficiently small amount of "noise", and $q \gg p \in \mathbb{Z}$ is a random large number. Decryption runs by "removing" the noise using modulo reductions. Namely, we evaluate $[[c]_p]_2$, where $p \in \mathbb{Z}$ is the secret key[6]. The first modulo reduction $[\cdot]_p$ allows one to remove the pq term, and the second one $[\cdot]_2$ to remove the noise $2r$. We are not interested at this point in investigating the security of the scheme. Nonetheless, observe its homomorphic properties:

5: Given several integers $x_1, x_2, \ldots, x_n$ that are near multiples of some hidden integer p, the goal is to find the hidden integer p.

6: We use the $[x]_p$ notation to indicate $x \bmod p$

- Homomorphic addition (XOR, $\oplus$): given two ciphertexts $c_1 := m_1 + 2r_1 + pq_1$ and $c_2 := m_2 + 2r_2 + pq_2$, we can compute their sum $c^+ := c_1 + c_2$, which will be correctly decryptable since:

$$\begin{aligned}[[c^+]_p]_2 &\equiv [[(m_1 + 2r_1 + pq_1) + (m_2 + 2r_2 + pq_2)]_p]_2 \\ &\equiv [[(m_1 + m_2) + 2(r_1 + r_2) + p(q_1 + q_2)]_p]_2 \\ &\equiv m_1 + m_2 \in \mathbb{Z}_2\end{aligned}$$

  .

  Notice that the amount of noise – namely $2(r_1 + r_2)$ – has roughly doubled.
- Homomorphic multiplication (AND, $\wedge$): given two ciphertexts $c_1 := m_1 + 2r_1 + pq_1$ and $c_2 := m_2 + 2r_2 + pq_2$, we can compute their multiplication $c^\times := c_1 \cdot c_2$, which will be correctly decryptable:

$$\begin{aligned}[[c^\times]_p]_2 &\equiv [[(m_1 + 2r_1 + pq_1) \cdot (m_2 + 2r_2 + pq_2)]_p]_2 \\ &\equiv [[(m_1 \cdot m_2) + 2 \cdot \text{noise} + p \cdot \ldots]_p]_2\end{aligned}$$

We avoid to show the whole noise term, but its amount is roughly squared after a multiplication.

Now, ignoring the eventual noise, an unlimited number of computations could be performed without issues, but then the cryptosystem would not be secure anymore[7]. Therefore, we must find a way to deal with the noise. Notice that a ciphertext can be correctly decrypted as long as its noise r is smaller than $p/2$ – if not, we have no guarantees that the message can be correctly recovered (see Example 1).

7: For instance, encryptions of zero will always have the same parity

Example 1 (*Noise in noisy schemes*)

Consider a ciphertext encrypting $m = 0$. If $p = 139$ and the noise is small, e.g., $r = 3$, then the ciphertext can be decrypted:

$$[[c]_p]_2 \equiv [[6 + pq]_p]_2 \equiv [6]_2 \equiv 0.$$

On the other hand, by picking, e.g., $r = 79$, the decryption fails:

$$[[c]_p]_2 \equiv [[158 + pq]_p]_2 \equiv [[19 + pq]_p]_2 \equiv [19]_2 \equiv 1.$$

Many arguments in subsequent chapters will heavily rely on this "decryption correctness" constraint. An intuition is that the amount of noise $2r$ must not affect the modulo reduction by p. Indeed, if $r > p/2$, then the total noise will be $2r > p$, randomizing the parity of the decryption, changing de facto the message m.

This naturally introduces a strong constraint: we can perform both additions and multiplications, but with some limitations. If the computation to be performed is known in advance, one could fix a sufficiently large set of parameters such that the probability that the noise exceeds the threshold is negligible. We will refer to this type of schemes as *leveled homomorphic encryption* schemes (LHE). Notice that, differently from a somewhat homomorphic encryption, one could theoretically evaluate any arbitrary circuit[8] by fixing a suitable set of parameters, although this approach quickly becomes unpractical.

8: With the term "circuit", we refer to a sequence of arithmetic or boolean operations.

1.1.3 Fully homomorphic encryption

We are now ready to introduce the fully homomorphic encryption (FHE) primitive, which allows one to evaluate circuits of any depth – without adapting the set of parameters – over some encrypted data. Currently, the only way that we know

today to achieve FHE is due to Gentry [Gen09b]. In particular, the proposed solution is to perform a *bootstrapping* procedure: the required ingredients are a SWHE scheme and a shallow decryption circuit. The abstract idea is to reduce the noise in a ciphertext by evaluating its decryption circuit as a homomorphic circuit, using an encrypted version of the secret key, called the *bootstrapping key*.

[Gen09b]: Gentry, *'Fully homomorphic encryption using ideal lattices'* . In: STOC '09

The original formulation. In its original formulation, bootstrapping was performed by tweaking the decryption circuit and by requiring an extra assumption. In particular, in the first FHE construction (refer to [Gen09a]), the decryption circuit was too complex to be evaluated by homomorphic operations, so Gentry proposed to *squash* the decryption circuit, namely to reduce its complexity. The abstract idea was to let the server evaluate in clear a part of the decryption circuit, and to perform the remaining part via homomorphic operations (using the encrypted secret key). This approach required an extra assumption[9] so that the server could not learn anything from the first part of the decryption. Nonetheless, modern FHE schemes do not use squashing or extra assumptions, since their decryption circuits are already shallow.

[Gen09a]: Gentry (2009), *'A fully homomorphic encryption scheme', Ph.D. Thesis, Stanford University.*

9: About the hardness of solving the sparse subset sum problem, given some hints.

Bootstrapping a NAND gate. Let us consider the most basic situation: we have two ciphertexts c_a and c_b, encrypting two binary values a, b, respectively. We (somehow) compute their NAND[10] as $c_{\bar{\wedge}} := c_a \;\bar{\wedge}\; c_b$, and the resulting ciphertext will contain a larger amount of noise with respect to fresh ciphertexts. The question is: how can we reduce the noise? Gentry's intuition was to encrypt the secret key under the public key – notice that this requires an additional assumption, i.e., *circular security* [MV24] – and to use the encrypted secret key to evaluate the decryption circuit of $c_{\bar{\wedge}}$. The evaluation of the decryption circuit will reduce the amount of noise in the ciphertext, enabling to perform another NAND operation.

10: Remark that NAND is a complete Boolean operator so it can be used to express any Boolean circuit

[MV24]: Micciancio and Vaikuntanathan, *'SoK: Learning with Errors, Circular Security, and Fully Homomorphic Encryption'* . In: PKC '24

Say that the parameters of the SWHE have been set to support up to ℓ operations, and the decryption circuit requires $\ell - 1$ operations, then we obtain a FHE scheme! This is roughly what was proposed in the FHEW scheme [DM15]. The objective of the latter is to bootstrap a ciphertext under a scheme $\mathcal{E}$ by (i) re-encrypting the ciphertext under a new (SWHE) scheme $\mathcal{E}_b$, (ii) evaluate the decryption circuit homomorphically and (iii) change the key back to the original scheme $\mathcal{E}$. If the scheme $\mathcal{E}_b$ well fits the decryption circuit of $\mathcal{E}$, we obtain a very efficient

[DM15]: Ducas and Micciancio, *'FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second'* . In: EUROCRYPT '15

and elegant approach to bootstrap. This (abstract) framework is summarized in Figure 7.

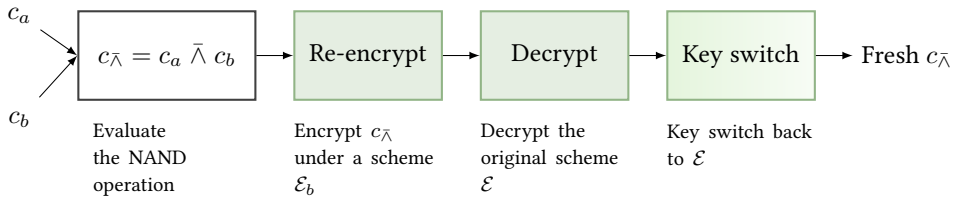


Figure 7: High-level overview of a simple bootstrapping framework inspired from [DM15]. Green boxes refer to $\mathcal{E}_b$, while the others to $\mathcal{E}$

Today, we mainly have two approaches to perform a bootstrapping:

- Fast-bootstrapping (or DM/CGGI-like): this refers to the just presented approach. A bootstrapping is usually run after each NAND operation (in more advanced constructions such as TFHE [CGGI20], after each multiplication) using another scheme $\mathcal{E}_b$ that is optimized to evaluate the decryption circuit of $\mathcal{E}$.
- Leveled bootstrapping (or BGV-like): although looking like an oxymoron, there are schemes that allow to perform many operations, and are also bootstrappable[11], so in this case the framework idea looks seemingly simpler, i.e., $\mathcal{E} = \mathcal{E}_b$. As an example, we can find the BGV [BGV12] scheme. Since these schemes do not fully focus on the bootstrapping – in fact it is not always required – this procedure is slower, although they have the ability of packing multiple values in a single ciphertext and to perform parallel Single instruction, multiple data (SIMD) operations.

[CGGI20]: Chillotti, Gama, Georgieva, and Izabachène (2020), *'TFHE: Fast Fully Homomorphic Encryption Over the Torus'* . In: Journal of Cryptology

11: They can evaluate their own decryption circuit

[BGV12]: Brakerski, Gentry, and Vaikuntanathan, *'(Leveled) fully homomorphic encryption without bootstrapping'* . In: ITCS '12

We refer the curious reader to [AP23] for more details about the different approaches to bootstrapping. A more in-depth and formal analysis of FHE, along with the various techniques to perform a bootstrapping, will be given in the next chapter.

1.2 The foundations of fully homomorphic encryption: lattices

All fully homomorphic encryption schemes in use today are constructed from standard lattice-based cryptographic

assumptions. A lattice is a seemingly simple mathematical object that can be described by a *basis*, that is a set of linearly independent vectors. Consider, for instance, the identity matrix. By picking all the integer combinations of the basis vectors[12], we obtain a set of points that corresponds to the lattice, refer to Figure 8.

12: We assume column vectors

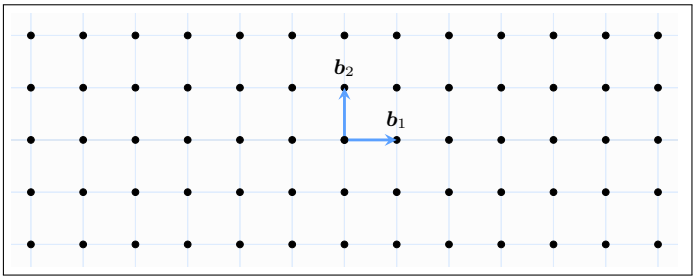


Figure 8: The simple lattice $\mathbb{Z}^2$ generated by the two dimensional identity matrix

Historically, lattices have been object of study by many famous mathematicians such as Lagrange, Gauss and Minkowski. In cryptology, they first appeared as tools to break cryptographic schemes, e.g., Shamir used them [Sha84] to break the Merkle-Hellman knapsack scheme, Coppersmith [Cop97] to break some low exponent RSA instances. They have been used as tools to *break* cryptosystems, not to build them – until 1996, when the breakthrough result by Ajtai [Ajt96] connected the average-case to worst-case hardness of some lattice problems.

[Sha84]: Shamir (1984), *'A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem'* . In: IEEE Transactions on Information Theory

[Cop97]: Coppersmith (1997), *'Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities'* . In: Journal of Cryptology

[Ajt96]: Ajtai, *'Generating hard instances of lattice problems (extended abstract)'* . In: STOC '96

[Reg05]: Regev, *'On lattices, learning with errors, random linear codes, and cryptography'* . In: STOC '05

As we will see later, most of the FHE schemes are based over some hard problems defined over a very specific family of lattices, called q-ary lattices. In particular, FHE schemes are secure as long as finding short vectors in such lattices is computationally hard. In standard constructions, the connection between lattices and FHE schemes has a name: Learning with errors (LWE) [Reg05], what will be discussed later. Some good properties of lattice-based cryptosystems are:

- Natural additivity (i.e., adding two ciphertexts gives another ciphertext)
- Simple decryption circuit (i.e., rounded inner product)
- High efficiency as they allow to naturally work with integer values modulo some (relatively small) q.

On the other hand, all lattice-based schemes rely on *errors*, just like in the previously presented DGHV scheme. One of the main contributions of this thesis (i.e., [LRvW25]) is a purely lattice perspective of FHE schemes, allowing to define the main ingredients required to build such a scheme – and to

have some geometrical intuitions on how these schemes work. A more formal and rigorous introduction to lattices will be given in Section 2.1.

1.3 Applications of fully homomorphic encryption

Fully homomorphic encryption is an essential tool towards completely private Internet communications; prominent and emerging applications include the following fields.

Privacy-preserving machine learning (PPML). Nowadays is becoming more and more practical to run machine learning (ML) models over encrypted data. The idea is somewhat simple, a ML model can be seen as a circuit which can be rewritten using FHE operators, so that encrypted inputs can be given to it, obtaining private outputs that can be seen in clear by the data owner only. We can find examples of both training [Mon+24] and inference from encrypted data. The number of possible use-cases of this technique is large, for example we could have systems to privately work on personal medical data [WNK20], on people face [AB25] and voice recognition [Ngu+25]. On a more sensitive level, there are studies (e.g., from European union [ML25]) aimed to use FHE to securely detect child abuse material without affecting the end-to-end encryption security of messaging applications, or to merge personal medical data between different hospitals and patients [Fro+21].

Private information retrieval (PIR). This type of protocols allow a user to retrieve a record from a database without revealing which item was requested. As an example, we refer to a sample implementation[13] of [MW22], where a Wikipedia service is offered. The user is able to write a query, which is encrypted and sent to the server. The latter is able to return to the user a web page corresponding to the sent key words, without knowing which page was requested. PIRs are evolving towards more efficient and faster constructions [MCR21; Col+23; Li+24]. Notice that these are not purely theoretical proposals as, for instance, Apple implemented[14] a PIR protocol to retrieve metadata (logo and information) about emails and incoming phone calls, or to check whether a requested URL has been classified as adult content.

13: Available at spiral-wiki.com

14: Refer to machinelearning.apple.com/research/homomorphic-encryption

1.4 Contribution of the thesis

During my Ph.D., I have been interested in problems concerning FHE from two points of view: constructions and algorithms. The contributions are summarized in Figure 9. In

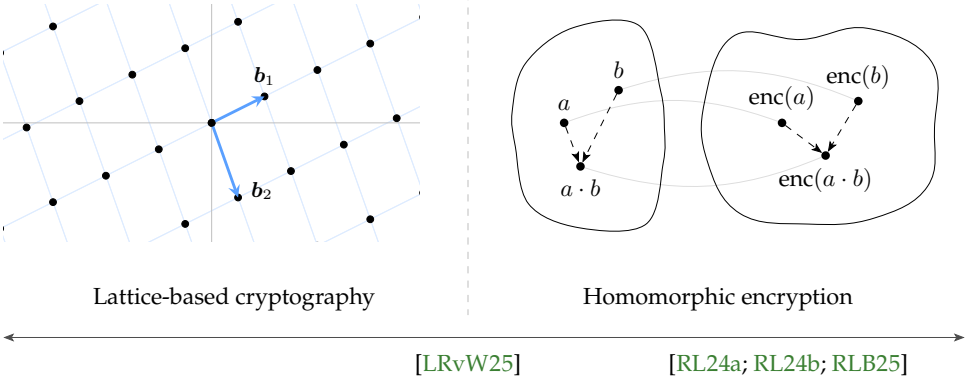


Figure 9: Summary of the topics and contributions that will be discussed in this thesis. The works are positioned in a spectrum that goes from lattice-based cryptography to homomorphic encryption

particular, [LRvW25] can be positioned as in between lattice-based cryptography and homomorphic encryption, while [RL24a; RL24b; RLB25] are pure FHE applications.

[LRvW25]: Beyond LWE: a Lattice Framework for Homomorphic Encryption.[15]

15: *Co-authored with A. Leporati and W. van Woerden, currently under review at PKC '26*

This paper deals with homomorphic encryption constructions from a purely lattice perspective. It tries to answer the following question.

Question 1

Which are the minimum lattice requirements to build an FHE scheme? Which hardness assumptions can we use?

Since LWE can be seen as a lattice problem, in this work we generalize some standard FHE schemes (in particular [BGV12; GSW13]) from a purely lattice perspective. In other words, our aim is to better understand which are the *lattice properties* that a scheme should have in order to be fully homomorphic. It turns out that the minimum requirements are the following:

- a short vector $s \in \mathcal{L}$ in the primal lattice $\mathcal{L}$;

[BGV12]: Brakerski, Gentry, and Vaikuntanathan, *'(Leveled) fully homomorphic encryption without bootstrapping'* . In: ITCS '12

[GSW13]: Gentry, Sahai, and Waters, *'Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based'* . In: CRYPTO '13

- an ambient space $\mathcal{D}$ for the dual lattice such that $\mathcal{D} \subset \mathcal{L}^*$ where small vectors can be sampled efficiently;
- a public description of the ambient space that does not allow for trivial decoding of Bounded distance decoding (BDD) targets.

Notice that such ingredients are (implicitly) present in LWE constructions, although by defining them explicitly, we open the doors to other possible hardness assumptions. In particular, the Lattice isomorphism problem (LIP) assumption [DvW22] can be used to construct a BGV-like/GSW-like scheme starting from these ingredients, and we will sketch a way to do so.

[DvW22]: Ducas and van Woerden, *'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'* . In: EUROCRYPT '22

[RL24a]: Encrypted Image Classification with Small Memory Footprint using Fully Homomorphic Encryption[16]

16: *Co-authored with A. Leporati, published in International Journal of Neural Systems*

Most of the already available solutions to classify encrypted images either require a large amount of memory (around 100 GB) or have low accuracy.

Question 2

Can we classify encrypted images with the same accuracy as the plain model without requiring a large amount of memory?

In this paper we use the CKKS scheme [CKKS17] to infer from a residual neural network (ResNet) [He+16a] with encrypted images. In particular, we consider the CIFAR-10 dataset, since other works [Lee22; KG23a; Ben+23a] can be used as reference benchmarks. We can position our work as illustrated in Figure 10.

[CKKS17]: Cheon, A. Kim, M. Kim, and Song, *'Homomorphic Encryption for Arithmetic of Approximate Numbers'* . In: ASIACRYPT '17

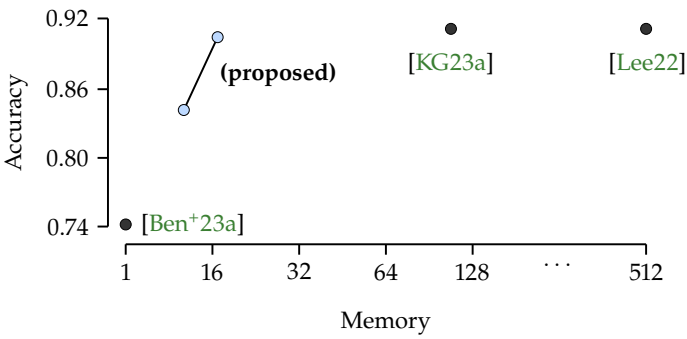


Figure 10: Framing [RL24a] results with the state-of-the-art proposals (at the time of the publication) with respect to accuracy and memory requirements. We propose two different solutions depending on the chosen set of parameters

Our solution consists in recycling *automorphism* keys, which are large objects required to perform rotations over the ciphertexts slots. In particular, we were able to reduce the number of

required rotation keys to a small constant value that does not depend on the size of the filter, being still able to perform practical convolutions. The experiments, easily replicable with our open-source code, show that our method allows one to obtain a decent amount of accuracy (comparable to [KG23a; Lee22]) while requiring orders of magnitude less memory. Luckily, this work has been used as a building block for different frameworks, among which we refer to, e.g., Prismo [Nju+25], which presents a recommendation system as an online web application.

[Nju+25]: Njungle, Jahns, Mastromauro, Kayang, et al. (2025), Prismo: A Decision Support System for Privacy-Preserving ML Framework Selection (arXiv paper 2510.09985)

[RL24b]: Transformer-based Language Models and Homomorphic Encryption: an Intersection with $\text{BERT}_{\text{tiny}}$[17]

17: *Co-authored with A. Leporati, published in the proceedings of IWSPA '24*

In the last year, the rise of language models is having a strong impact in our society. Despite their outstanding performance, they bring countless issues in terms of security and privacy when offered as online services by private companies. The main building block of such models is the Transformer architecture [Vas+17a]; so we tried to answer the following question.

[Vas+17a]: Vaswani, Shazeer, Parmar, Uszkoreit, et al., *'Attention is All you Need'* . In: NeurIPS 2017

Question 3

Can we efficiently evaluate Transformer architectures under FHE? Which are the main bottlenecks?

Since previous works proposed only hybrid architectures, we wanted to define a basic benchmark for the simplest possible model based on Transformer, namely $\text{BERT}_{\text{tiny}}$. In particular, we have considered a fine-tuned version, specifically on the SST-2 dataset[18]. Results, in terms of runtime and number of tokens, are presented in Figure 11, and confirm the linear complexity of the FHE circuit.

18: A dataset for predicting sentiment from long movie reviews

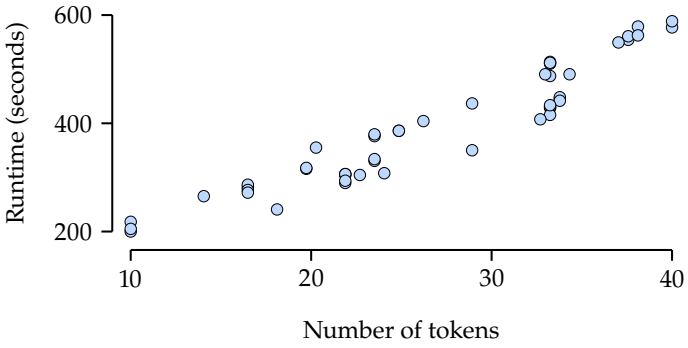


Figure 11: Framing [RL24b] results with respect to accuracy and memory requirements, in the literature at the time of the publication

The main bottleneck is given by the depth of the circuit, which in turn implies a large number of bootstrapping operations.

[RLB25]: Lightweight Sorting in Approximate Homomorphic Encryption[19]

19: *Co-authored with A. Leporati and S. Basile, currently under review in IACR Communications in Cryptology*

This paper deals with a crucial algorithmic primitive: sorting numbers. Under FHE, this is far from being an easy task as comparison operations imply the evaluation of nonlinear functions, which in our approximate setting must be approximated using polynomials.

Question 4

Can we improve current sorting algorithms under CKKS?

In particular, our objective has been to improve two already available approaches: permutation-based sorting [MEHP25] and network-based sorting [Hon+21]. Concerning the first, we were able to reduce the depth of the circuit from 65 multiplicative levels down to 20, without decreasing the precision of the output results; the core idea of our solution is to replace the evaluation of very complex polynomials with some lighter ones which give the same result. At the core of our solution there is the $\text{sinc}(x) = \sin(\pi x)/\pi x$ function, which can be used to verify whether two (integer) values a, b are equal by evaluating $\text{sinc}(a - b)$. Secondly, we improved the network-based approach by proposing an algorithm that evaluates a complete layer of a sorting network (see Figure 12) in a single polynomial approximation of the $\text{ReLU}(x) = \max(0, x)$ function, which implies orders of magnitude faster runtimes than in [Hon+21] (where comparisons are performed using approximations of the $\text{sgn}(x)$ function).

[MEHP25]: Mazzone, Everts, Hahn, and Peter, *'Efficient Ranking, Order Statistics, and Sorting under CKKS'* . In: USENIX '25

[Hon+21]: S. Hong, S. Kim, Choi, Y. Lee, et al. (2021), *'Efficient Sorting of Homomorphic Encrypted Data With k-Way Sorting Network'* . In: IEEE Trans. on Information Forensics and Security

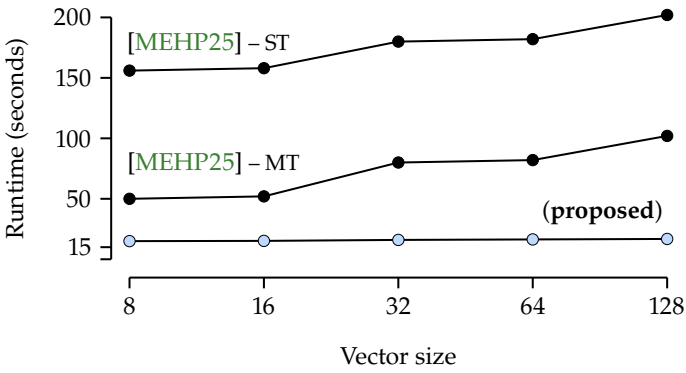


Figure 12: Comparing permutation-based sorting with [MEHP25]. The first [RLB25] results with respect to accuracy and memory requirements, in the literature at the time of the publication

Building the groundwork | 2

Beginnings are such delicate times.

(F. Herbert, Dune)

In this chapter we introduce all the mathematical and cryptographic concepts that are, directly or indirectly, required in better understanding the subsequent parts of the thesis.

Some general preliminaries. We start by defining the basic notation used throughout the thesis – if not stated otherwise. Lowercase letters are used to indicate numbers (e.g. a), bold-italic lowercase letters to indicate vectors (e.g. $\boldsymbol{v}$), and uppercase letters are used to indicate matrices (e.g. A). Vectors are considered as column vectors, the i-th column of a matrix A is referred to as $\boldsymbol{a}_i$ and the i-th element of a vector $\boldsymbol{v}$ as v_i. We use, except if stated otherwise, an indexing starting from 1, i.e., $\boldsymbol{a}_1$ is the first element of $\boldsymbol{a}$. The symbols $\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$ refer to the sets of natural, integer, real and complex numbers. We use the notation $\mathbb{Z}[X], \mathbb{Q}[X], \mathbb{R}[X], \mathbb{C}[X]$ to refer to the set of polynomials with integer, rational, real and complex coefficients, respectively.

A *group* is a set G with a binary operation $\circ$ that is associative, has an identity element e (typically 1), and such that every $a \in G$ has an inverse $a^{-1} \in G$, namely $a \circ a^{-1} = e$. A *ring* is a set R with two binary operations $+$ and $\cdot$, where $(R, +)$ is an abelian group, multiplication is associative, and multiplication distributes over addition. A *field* is a commutative ring F with unity $1 \neq 0$ in which every nonzero element has a multiplicative inverse. When using the term *ring*, we assume a commutative ring, i.e., such that for every $a, b \in \mathcal{R}$, it holds that $a \cdot b = b \cdot a$.

A vector space over a field (usually $\mathbb{R}$) is a set where vectors can be added together and multiplied by a scalar from the field. The symbol $\mathbb{Z}_q$ refers to the cyclic group $\mathbb{Z}/q\mathbb{Z}$ of integers modulo q, which for convenience we may identify with $[-q/2, \ldots, q/2) \cap \mathbb{Z}$ or $[0 \ldots, q) \cap \mathbb{Z}$, depending on the context. The general linear group of invertible matrices with integer (or real) entries of dimension m is referred to as $\mathrm{GL}_m(\mathbb{Z})$ (or $\mathrm{GL}_m(\mathbb{R})$, respectively).

The notation $a \sim \chi$ indicates some value a sampled from the probability distribution χ. The notation $[a]_q : \mathbb{Z} \to \mathbb{Z}_q$ indicates the modulo reduction $a \bmod q$. The $=$ symbol indicates an equality, while $\equiv$ an equivalence.

The notation $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{m} a_i b_i$ indicates the inner product in $\mathbb{R}^m$. The notation for the norms is $\|\boldsymbol{x}\|_n = (\sum_i |x_i|^n)^{1/n}$, if not specified the Euclidean norm is assumed, namely $\|\boldsymbol{x}\| := \|\boldsymbol{x}\|_2$. Lastly, the infinity norm is defined as $\|\boldsymbol{x}\|_\infty = \max_{1 \le i \le n}(|x_i|)$. The norm of a matrix is defined as the length of its longest vector $\|M\| = \max_i \|\boldsymbol{m}_i\|$.

A *module* is a generalization of a vector space where the field is replaced by some ring. For example, the $\mathbb{Z}$-module is the set of (integer) vectors that can be added together and can be scaled by integer values.

The notation $\deg(p)$ refers to the degree of the polynomial $p(X)$.

We use the λ symbol to refer to the security parameter. Informally, we say that if a set of parameters for a scheme has λ security bits, then the best known attack requires complexity roughly $O(2^\lambda)$. We refer to [MW18].

[MW18]: Micciancio and Walter, *'On the Bit Security of Cryptographic Primitives'* . In: EUROCRYPT '18

We say that a decision problem A reduces – as a Turing reduction – to some decision problem B and we indicate it with $A \le B$ if, by solving B, one could also solve A with a polynomial number of calls to an oracle solving B. Karp reductions are considered as a special case where the oracle is called only once. By abuse of notation, we might use it to also for search problems that do not necessarily fit into complexity classes but rather in hardness assumptions. For example, in the latter case, one would read $\mathrm{GapSVP}_\gamma \le \mathrm{LWE}$ as 'an algorithm that efficiently solves LWE implies an algorithm that efficiently solves GapSVP_γ as well'.

Since the focus of this thesis is not to precisely define hardness of problems, we will use the big-O notation ignoring "soft" differences between O and $\tilde{O}$, namely we ignore logarithmic factors.

Rise and fall of knapsack. The foundations of modern FHE schemes can be dated back to the late 70s. From one side, we had some efforts aimed to study homomorphic properties of cryptosystems [RAD78], on the other we had a rising field that aimed to create schemes based on the worst-case of known NP-complete problems [Dif88] – most public keys schemes at the time were based on integer factorization that is not,

[RAD78]: R. L. Rivest, Adleman, and Dertouzos (1978), *'On data banks and privacy homomorphisms'* . In: Foundations of secure computation

[Dif88]: Diffie (1988), *'The first ten years of public-key cryptography'* . In: Proceedings of the IEEE

for instance, a NP-complete problem[1]. Although even today there are no examples of cryptographic constructions that are hard merely assuming P $\neq$ NP, there are ways to get somehow close to NP-hardness. One example is given by hard lattice problems, but before that, we introduce their "grandparents", *knapsack problems*.

1: Its decision version (i.e., does n have a prime factor $< a$?) is in NP $\cap$ coNP.

The idea is somehow simple: the field of complexity theory already has some strong results about the hardness of solving certain problems – in the worst case. What we would like to do is therefore to build a cryptosystem above these results. One of the first attempts is based on the *Subset sum* problem, known for being an NP-complete problem, in its decision variant.

Definition 1 (*Subset sum*)

Given some positive weights $\boldsymbol{a} \in \mathbb{Z}^n$ and a target value $t \in \mathbb{Z}$, find $\boldsymbol{x} \in \{0,1\}^n$ such that $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = t$.

Recall that NP-completeness is defined with respect to the *worst-case* of a problem, namely we have the confidence that finding an algorithm that solves all the instances is a pretty hard task. Nevertheless, this is a weak argument for cryptography, as we would like to generate instances that are "always" hard to solve, except with negligible probability. NP-completeness is indeed not a sufficient condition to build a scheme, and we usually require average-case complexity. A very interesting discussion on this topic is given by Bogdanov and Trevisan [BT06].

[BT06]: Bogdanov and Trevisan (2006), *'Average-Case Complexity'* . In: Foundations and Trends in Theoretical Computer Science

One of the first schemes based on Subset sum is the Merkle-Hellman scheme [MH78]. When considering a hard problem for a public-key scheme, one of the main issues is about how to efficiently generate an instance containing a (secret) trapdoor[2], but at the same time hard to be solved without having access to it. In particular, the subset sum weights $\boldsymbol{a} \in \mathbb{Z}^n$, as defined in [MH78], are in a so-called superincreasing sequence[3], and this is a requirement for an efficient decryption. Nevertheless, this means that [MH78] is not based on worst-case subset sum, but only on some specific instances. The public key of the scheme is given by $\boldsymbol{a}$, mixed with some random permutation, modulus reduction and a random multiplier.

[MH78]: R. Merkle and Hellman (1978), *'Hiding information and signatures in trapdoor knapsacks'* . In: IEEE Transactions on Information Theory

2: Simply put, a solution to that instance

3: A sequence $(a_1, a_2, \ldots, a_n)$ where each $a_i > \sum_{j=1}^{i-1} a_j$ for all i

This superincreasing sequence property was exploited a few years later by Shamir [Sha84] to completely break the scheme in polynomial time. This attack made knapsack-based cryptosystem not particularly attractive in the following years, although we can find examples of schemes which are not yet fully broken, e.g., [CR88].

[Sha84]: Shamir (1984), *'A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem'* . In: IEEE Transactions on Information Theory

For a broader survey, refer to [Odl90], which inspired the title of this paragraph. Better fortune smiled to knapsack's grandchildren – codes and lattices, which we can say are on their "rise" phase. For the rest of this thesis, we will focus on the latter.

[Odl90]: Odlyzko, *'The Rise and Fall of Knapsack Cryptosystems'* . In: Proceedings of Symposia in Applied Mathematics

2.1 Lattices

Almost everything in this thesis deals, directly or indirectly, with *lattices*: some curious and seemingly simple mathematical objects. These have been well-known and studied for many years by famous mathematicians such as Lagrange, Gauss, Voronoi and Minkowski, among others.

In the field of cryptology, they have been used for many years as tools to break schemes (e.g., some RSA instances [Cop97], or knapsack-based [Sha84]), but not to build them. The seminal work by Ajtai [Ajt96], which connected worst-case to average-case hardness for certain lattice problems, and the subsequent cryptosystem [AD97], marked on the other hand the beginning of a "lattice renaissance" in the field of cryptography. The work by Ajtai had, as a a result, an evidence that solving certain problems over random instances is as hard as solving them in the worst case. From a cryptographic perspective, breaking a cryptosystem on average would imply a solutions to all instances of the problem – a much harder task.

[Ajt96]: Ajtai, *'Generating hard instances of lattice problems (extended abstract)'* . In: STOC '96

[AD97]: Ajtai and Dwork, *'A public-key cryptosystem with worst-case/average-case equivalence'* . In: STOC '97

Different resources inspired the writing of this chapter, including, but not limited to, [MG02; Mic21; Pei22; Vai21].

Definition 2 (*Lattice, in general*)

A lattice $\mathcal{L} \subset \mathbb{R}^n$ *is a discrete subgroup of* $\mathbb{R}^n$ *that is closed under subtraction.*

Hence a *lattice* is a set that is (i) closed under subtraction[4] (i.e., for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$, it holds that $\boldsymbol{x} - \boldsymbol{y} \in \mathcal{L}$), and (ii) is discrete, namely there always exists an $\varepsilon > 0$ such that any two distinct lattice points $\boldsymbol{x} \neq \boldsymbol{y}$ are at distance at least $\|\boldsymbol{x} - \boldsymbol{y}\| \geq \varepsilon$.

4: Addition can be expressed using subtractions as $\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{x} - (-\boldsymbol{y})$

Nevertheless, this definition can be a bit abstract, and we will begin by considering lattices generated by a so-called *basis*.

Definition 3 (*Lattice, given a basis*)

Let $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$ *be linearly independent vectors forming a so-called* basis $B := (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^n$. *The* lattice *generated by*

B is defined as

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_k) = \mathcal{L}(B) = \Big\{ \sum_{i=1}^{k} z_i \mathbf{b}_i \;\Big|\; z_i \in \mathbb{Z} \Big\}. \tag{2.1}$$

The set $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is called a basis *of the lattice. The integer k is the* rank *of the lattice, and if $k = n$, we call it a* full-rank lattice *in $\mathbb{R}^n$.*

To describe a lattice, which is an infinite set of points, it is therefore sufficient to provide a so-called *basis* $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_n\} \in \mathbb{R}^n$, and generate all the possible integer combinations of these vectors – note that, if not stated otherwise, we will mostly deal with full-rank square bases. We also introduce the concept of *span* of a lattice.

Definition 4 (*Span of a lattice*)

For any lattice $\mathcal{L}(B)$, its span *is given by:*

$$\text{span}(\mathcal{L}(B)) := \{B\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^n\}. \tag{2.2}$$

We might also indicate $\text{span}(B)$ or $\text{span}(\mathcal{L})$ for simplicity. The difference between (2.1) and (2.2) is that, in the span we consider arbitrary real coefficients to define the space, while in the definition of the lattice we simply pick the integer combinations.

The most basic lattice is given by the identity matrix, namely $B = I_n$ for a dimension $n > 0$, refer to Figure 13 for the two dimensional case.

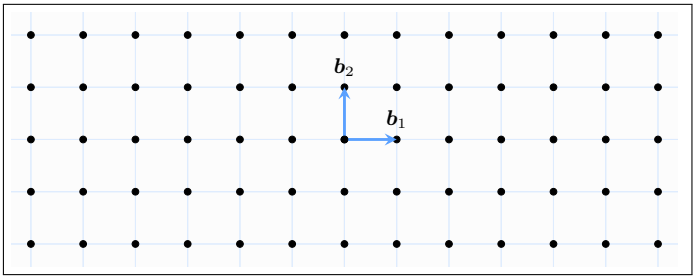


Figure 13: The simplest lattice $\mathbb{Z}^n$, with $n = 2$, generated by two basis vectors $\boldsymbol{b}_1 = (1, 0)$ and $\boldsymbol{b}_2 = (0, 1)$

In general, we can consider the lattice $\mathcal{L}(B)$ as the application of a linear transformation to $\mathbb{Z}^n$, namely $\mathcal{L}(B) = B \cdot \mathbb{Z}^n$. In Figure 14, we illustrate the lattice generated by the two dimensional basis $B = \begin{bmatrix} 1 & 1/2 \\ 1/2 & -\sqrt{2} \end{bmatrix}$, consider the basis vectors as the columns of the matrix.

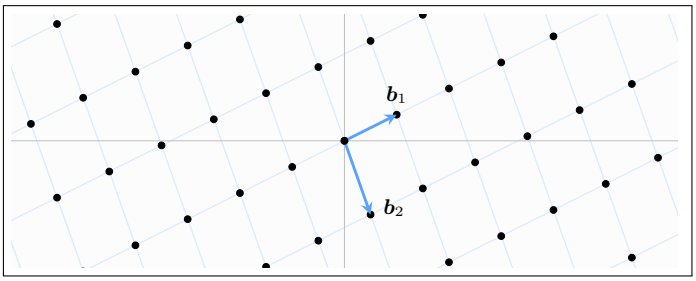


Figure 14: A different lattice generated by applying a linear transformation B to $\mathbb{Z}^n$

If two bases B, B' differ by a unimodular transformation $U \in \mathrm{GL}_n(\mathbb{Z})$ (i.e., integer and such that $\det(U) = \pm 1$) as $B' = B \cdot U$, then the two lattices $\mathcal{L}(B)$ and $\mathcal{L}(B')$ refer to exactly the same set of points. For example, in Figure 15 we present the same lattice as the one presented in Figure 14, but generated by a different basis.

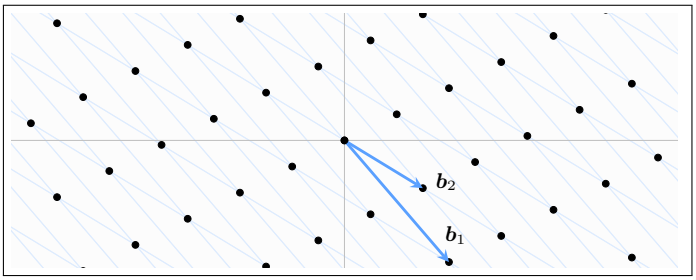


Figure 15: A different lattice generated by applying a linear transformation B to $\mathbb{Z}^n$

Notice that, differently from the previous case, the "grid" generated by this basis is not orthogonal anymore.

Lemma 1 (*Equivalence between bases*)

Two lattice bases $B, B' \in \mathbb{R}^n$ are said to be equivalent *if and only if $B' = B \cdot U$ for some unimodular matrix $U \in GL_n(\mathbb{Z})$.*

Corollary 1 (*Equal lattices*)

Two lattices are equal *if their bases are equivalent.*

We can associate different numerical quantities to every lattice, starting from the *determinant*. We will formally define it later. Before, we spoil that it is easy to "visualize" this quantity since it refers to the volume of the *fundamental region* (sometimes referred to the *fundamental parallelepiped*), which is the (centered) parallelepiped spun by the basis vectors, as shown in Figure 16.

Definition 5 (*Fundamental region*)

For any lattice $\mathcal{L}(B)$, the fundamental region *associated to the*

lattice is the region

$$\mathcal{P} := B \cdot [0,1)^n \subset \mathbb{R}^n.$$

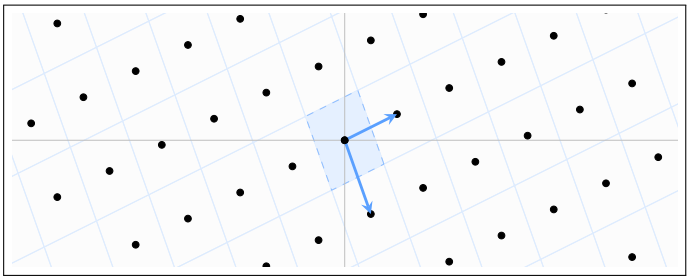


Figure 16: A lattice and its centered fundamental region indicated as a light blue area

Depending on the desired representation, one can also use the centered fundamental region (just like in the previous figure):

$$\mathcal{P} := B \cdot \left[\frac{1}{2}, \frac{1}{2}\right)^n \subset \mathbb{R}^n.$$

As we will better see later, both the standard fundamental region and the centered one serve as canonical sets of representatives for the quotient $\mathbb{R}^n/\mathcal{L}(B)$, differing only by a translation.

Going back to the determinant, it is possible to compute it by taking advantage of the *Gram-Schmidt orthogonalization* (GSO) process, formally defined as follows.

Definition 6 (*Gram-Schmidt orthogonalization*)

For any matrix $B \in \mathbb{R}^{m \times n}$ and its corresponding column vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$, the corresponding Gram-Schmidt vectors *$\tilde{\boldsymbol{b}}_1, \tilde{\boldsymbol{b}}_2, \ldots, \tilde{\boldsymbol{b}}_n \in \mathbb{R}^m$ are defined as:*

$$\tilde{\boldsymbol{b}}_i := \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \cdot \tilde{\boldsymbol{b}}_j \qquad (2.3)$$

$$\text{where } \mu_{i,j} := \frac{\langle \boldsymbol{b}_i, \tilde{\boldsymbol{b}}_j \rangle}{\langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle}.$$

The span of the basis vector $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ and the one of their Gram-Schmidt orthogonalized $\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n$ are the same, i.e., $\text{span}(B) = \text{span}(\tilde{B})$[5], but the vectors $\tilde{\boldsymbol{b}}_i$ are pairwise orthogonal – i.e., $\langle \tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_j \rangle = 0$. Therefore, we can compute the volume of the fundamental parallelepiped simply by computing the product between the norms of the GSO vectors.

5: With $\tilde{B}$ we refer to $\tilde{B} := \left(\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n\right)$

Definition 7 (*Determinant*)

For any lattice $\mathcal{L}(B)$, the determinant *associated to the lattice is given by*

$$\det(\mathcal{L}(B)) := \text{vol}(\mathcal{P}) = \prod_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|. \tag{2.4}$$

With $\text{vol}(\mathcal{P})$ we refer to the volume of the fundamental region. The determinant is a well-defined quantity of a lattice, in the sense that it does not refer to a specific basis. Even if the shape of the fundamental region changes after the application of some unimodular matrix U, its volume is preserved. A way of looking at the determinant is also in terms of the density of points. Larger determinants indeed define sparser lattices, and vice versa[6]. Remark, though, that the two bases B and $\tilde{B}$ do not necessarily generate the same lattice, as most lattices do not have a basis consisting of orthogonal vectors.

6: The determinant can indeed be seen as a "scaling factor"

Another way of computing the determinant is by evaluating the square root of the determinant of the Gram matrix $B^T B$, i.e.,

$$\det(\mathcal{L}(B)) = \sqrt{\det(B^T B)}. \tag{2.5}$$

Lastly, if we consider the case when the basis matrix B is square and nonsingular, then $\det(B) = \det(B^T)$, leading to

$$\sqrt{\det(B^T B)} = \sqrt{\det(B^T) \cdot \det(B)} = \det(B). \tag{2.6}$$

There are other useful quantities related to lattices which, differently from the determinant, are hard to be computed. For example, we will be very much interested in the computation of the *successive minima* of a lattice.

Definition 8 (*Successive minima*)

The i-th successive minimum λ_i *of a lattice $\mathcal{L}$ is the radius of the closed ball $B(0, \lambda_i)$ that contains exactly i independent nonzero lattice vectors.*

A representation of two successive minima λ_1, λ_2 is given in Figure 17. This looks like an easy task in two dimensions. Indeed, there is a pretty old algorithm attributed to Lagrange and Gauss that can be used to find λ_1 and λ_2 in polynomial time. Nonetheless, no efficient algorithm to find even only λ_1, for any dimension n, is known – even for polynomial approximation factors. However, we have some upper bounds on these quantities from Minkowski [Min96], whose early

[Min96]: Minkowski (1896), Geometrie der Zahlen

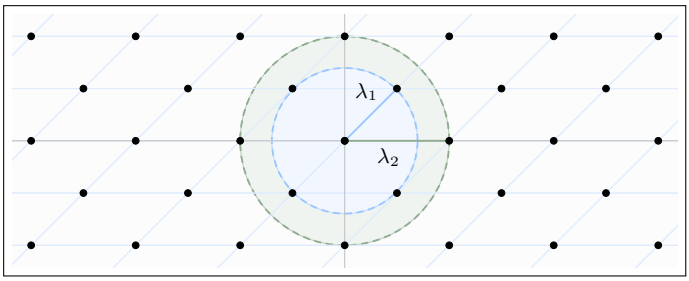


Figure 17: A two dimensional lattice and its two successive minima λ_1, λ_2

work laid the groundwork for much of modern geometry of lattices.

Theorem 1 (*Minkowski's first theorem*)

For any lattice $\mathcal{L}$ generated by a full-rank matrix B,

$$\lambda_1(\mathcal{L}) \leq \sqrt{n}(\det(\mathcal{L}))^{1/n}. \tag{2.7}$$

Notice that this result only ensures that a short vector exists, but the proof does not show a way to find it. A slightly more general bound is also given by the following result.

Theorem 2 (*Minkowski's second theorem*)

For any lattice $\mathcal{L}(B)$ generated by a full-rank matrix B,

$$\lambda_1(\mathcal{L}) \leq \Big(\prod_i \lambda_i(\mathcal{L})\Big)^{1/n} \leq \sqrt{n}(\det(\mathcal{L}))^{1/n}. \tag{2.8}$$

These two theorems can be proved starting from Blitchfeldt theorem [Bli14] and the Minkoswki's Convex body theorem (sometimes called Minkowski's First theorem), we refer the interested reader to [MG02, Section 1.3] for an exhaustive explanation and proofs.

[Bli14]: Blichfeldt (1914), *'A New Principle in the Geometry of Numbers, with Some Applications'* . In: Transactions of the American Mathematical Society

Gram-Schmidt orthogonalization and the first minimum of a lattice have a very strong relation.

Lemma 2 (*Lower bound on λ_1*)

For any lattice $\mathcal{L}(B)$, we have $\lambda_1(\mathcal{L}) \geq \min_i \|\tilde{\boldsymbol{b}}_i\|$

Proof. Consider a generic lattice point given by $B\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$. Set k to be the largest index such that $x_k \neq 0$. Now, take the scalar product between the lattice point and $\tilde{\boldsymbol{b}}_k$:

$$\langle B\boldsymbol{x}, \tilde{\boldsymbol{b}}_k \rangle = \sum_{i=1}^{k} \langle \boldsymbol{b}_i x_i, \tilde{\boldsymbol{b}}_k \rangle,$$

but since $\boldsymbol{b}_i$ and $\tilde{\boldsymbol{b}}_k$ are orthogonal for $i < k$ (so $\langle \boldsymbol{b}_i, \tilde{\boldsymbol{b}}_k \rangle = 0$), we have that

$$\sum_{i=1}^{k} \langle \boldsymbol{b}_i x_i, \tilde{\boldsymbol{b}}_k \rangle = \langle \boldsymbol{b}_k x_k, \tilde{\boldsymbol{b}}_k \rangle = x_k \langle \boldsymbol{b}_k, \tilde{\boldsymbol{b}}_k \rangle$$

by definition of GSO, we have that $\langle \boldsymbol{b}_k, \tilde{\boldsymbol{b}}_k \rangle = \langle \tilde{\boldsymbol{b}}_k, \tilde{\boldsymbol{b}}_k \rangle$, so:

$$\langle B\boldsymbol{x}, \tilde{\boldsymbol{b}}_k \rangle = x_k \|\tilde{\boldsymbol{b}}_k\|^2.$$

Now we can simply bound using Cauchy-Schwarz as

$$\|B\boldsymbol{x}\| \cdot \|\tilde{\boldsymbol{b}}_k\| \geq \|\langle B\boldsymbol{x}, \tilde{\boldsymbol{b}}_k \rangle \geq x_k \cdot \|\boldsymbol{b}_k\|^2.$$

Since $|x_k| > 0$, we divide by $\|\tilde{\boldsymbol{b}}_k\|$ to obtain $B\boldsymbol{x} \geq \|\tilde{\boldsymbol{b}}_k\|$. So any nonzero lattice vector $B\boldsymbol{x}$ has larger norm than the minimum of the GSO vectors of the basis. □

Hence, we have a way to set lower and upper bounds on the length of the shortest vector of a lattice. It is possible to notice, from Figure 17, that the two successive minima coincide with two basis vectors that generate the two dimensional lattice. This observation allows us to introduce the concept of *good basis* and *bad basis*. Consider the basis presented in Figure 18.

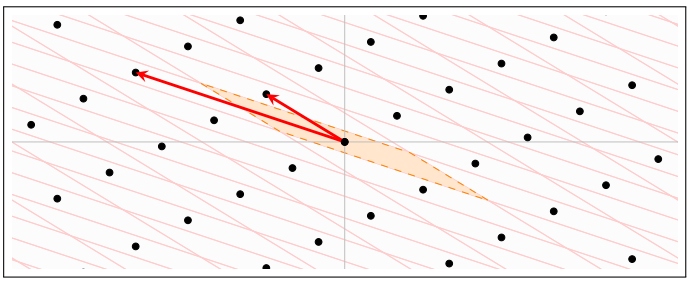


Figure 18: A lattice described by a bad basis: long and not orthogonal

Informally, a basis whose vectors are long and far from being orthogonal is considered a bad basis. We will see in the next subsection that solving certain problems – such as finding short vectors – seems very hard, given this bad description of $\mathcal{L}$. Conversely, if the basis consists of short and orthogonal vectors, then it is considered a good basis. Usually, a good basis allows one to efficiently solve some hard problems – think of the previous observation of the successive minima. A visual comparison between a good and a bad basis is given in Figure 19.

We will explore further the classification of good and bad

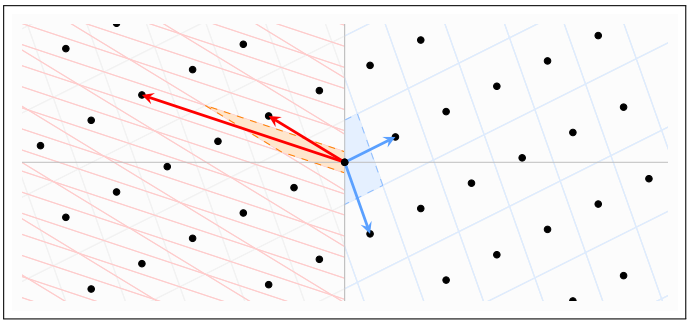


Figure 19: A lattice described, on the left, by a bad basis; on the right, by a good basis. From a cryptographic point of view, we will see how these are the attacker and the key owner perspectives, respectively

bases in Section 2.1.4.

Distances and covering radius. Before discussing hard problems, it is important to give a definition of the *distance* used in lattices.

Definition 9 (*Distance function*)

For any lattice $\mathcal{L}(B)$ *and target* $\boldsymbol{t} \in \text{span}(\mathcal{L}(B))$*, the* distance function *d is defined as*

$$d(\mathcal{L}(B), \boldsymbol{t}) = \min_{\boldsymbol{v} \in \mathcal{L}(B)} \|\boldsymbol{t} - \boldsymbol{v}\|. \tag{2.9}$$

In particular, the minimum of this function is 0, it can simply be obtained by picking $\boldsymbol{t}$ to be a lattice point. We will later see that, in general, computing the distance is far from being an easy task. Nonetheless, we have a way to bound the maximum distance between some target $\boldsymbol{t} \in \text{span}(\mathcal{L})$ and $\mathcal{L}$.

Definition 10 (*Covering radius*)

For any lattice $\mathcal{L}(B)$*, the* covering radius $\mu(\mathcal{L}(B))$ *is defined as:*

$$\mu(\mathcal{L}(B)) := \max_{\boldsymbol{x} \in \mathbb{R}^n} \min_{\boldsymbol{v} \in \mathcal{L}} \|\boldsymbol{x} - \boldsymbol{v}\|. \tag{2.10}$$

Consider that for any point $\boldsymbol{x} \in \mathbb{R}^n$, the value of $\min_{\boldsymbol{v} \in \mathcal{L}} \|\boldsymbol{x} - \boldsymbol{v}\|$ is the distance between $\boldsymbol{x}$ and the closest lattice point. The covering radius is the largest possible among these distances. We can visualize it as the largest possible radius of a sphere in the span of the lattice that does not touch any lattice point (refer to Figure 20). The name "covering radius" comes from the fact that, by putting these spheres around each lattice point, we will cover the entire spanned space, although with some overlappings (Figure 21). As the figure suggests, if some

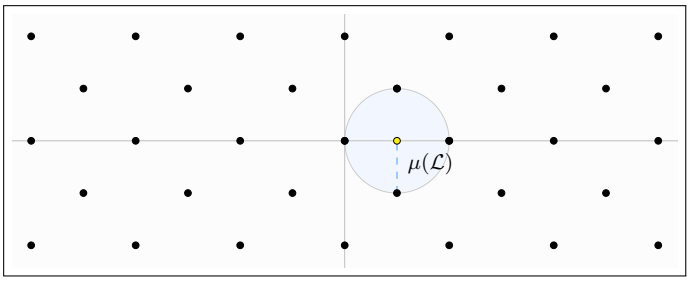


Figure 20: The "hexagonal" lattice and its covering radius

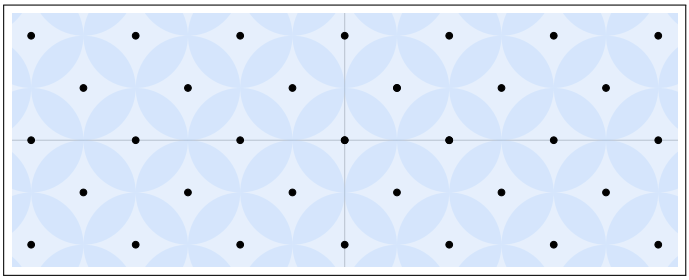


Figure 21: The "hexagonal" lattice and spheres around each lattice point of radius equal to the covering radius

target lies inside the overlapping zones, it is contained into (at least) two different spheres. If we want to uniquely decode some target using these spheres, we need another (in some sense "dual") definition.

Definition 11 (*Packing radius*)

For any lattice $\mathcal{L}(B)$, the packing radius $r(\mathcal{L}(B))$ *is defined as:*

$$r(\mathcal{L}(B)) := \frac{1}{2} \min_{\boldsymbol{v} \in \mathcal{L}(B)} \|\boldsymbol{v}\|$$

It is easy to see that this equals half of $\lambda_1(\mathcal{L}(B))$. In visual terms, the packing radius is the maximum radius $r > 0$ such that the spheres centered in each lattice point do not intersect (Figure 22). By taking advantage of this concept, we can define

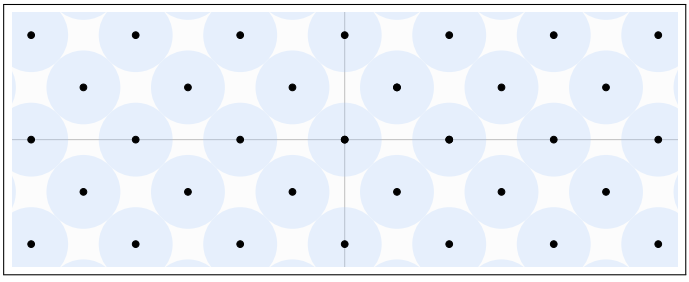


Figure 22: The "hexagonal" lattice and spheres around each lattice point of radius equal to the packing radius

uniquely decodable targets by simply considering targets that lie inside some sphere of radius at most $r(\mathcal{L})$.

A compact representation via fundamental regions. Lattices are infinitely large sets of points, so in order to study them from a computational point of view, we require a way to represent their structure and problems in a finite and discrete way. One of the most important functions that we use in lattices is the distance function. Notice though that this is periodic *modulo the lattice*. In particular, we can consider the fundamental region as a standard representative, and perform all the computations there. We give an intuition of the idea in Figure 23; notice that we can consider all the yellow points as equivalent, modulo the lattice – as they all have the same distance to the lattice.

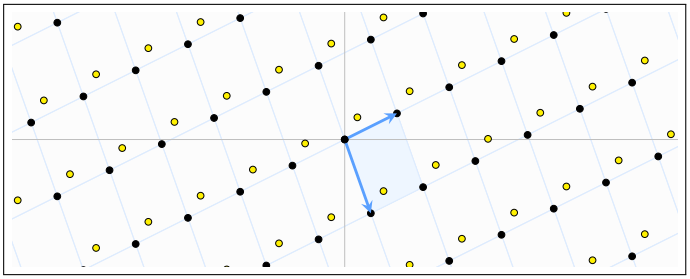


Figure 23: A lattice $\subset \mathbb{R}^n$ and some (yellow) points in $\mathbb{R}^n$. By working modulo the lattice, all the yellow points are equivalent since they have the same distance to the lattice

Therefore we reduce the lattice, that is an infinite set of points, to its fundamental region (which is anyway tiled throughout $\mathbb{R}^n$). Since $(\mathbb{R}^n, +)$ is an additive group, we can define $(\mathcal{L}, +)$ as an additive subgroup. Therefore, we can construct the quotient group $\mathbb{R}^n/\mathcal{L}$, and observe that the elements in this quotient group are all the cosets[7] of $\mathcal{L}$, namely

$$\mathbb{R}^n/\mathcal{L} = \{\mathcal{L} + \boldsymbol{t} : \boldsymbol{t} \in \mathbb{R}^n\}.$$

7: One can see a coset $\mathcal{L}+\boldsymbol{t}$ of $\mathcal{L}$ as simply the lattice translated by $\boldsymbol{t}$

In other words, we consider all of $\mathbb{R}^n$, and we identify vectors that differ by a lattice point. Any fundamental region of $\mathcal{L}$ gives a set of standard representatives, in fact

$$\mathbb{R}^n/\mathcal{L} \equiv B \cdot [0,1)^n = \mathcal{P}.$$

for any $B \in \mathbb{R}^n$ generating $\mathcal{L}$. In the case of Figure 23, instead of working with any yellow point in $\mathbb{R}^n$, we can simply consider the one inside the fundamental region. Notice that working modulo a lattice is the first requirement that allows us to study the hard computational problems under a complexity theory perspective. In fact, it allows to deal with $\mathcal{L}$ by only considering its "bounded" fundamental region $\mathcal{P}$ which contains only one representative for each coset of $\mathbb{R}^n/\mathcal{L}$ and has finite volume.

Notice that we are now able to consider a bounded number of lattice points in (i.e., the ones inside $\mathcal{P}$), but the underlying space in which $\mathcal{P}$ lies ($\mathcal{P} \subset \mathbb{R}^n$) is still dense, and not discretized. We have multiple ways to deal with this problem, for instance (i) use lattices that are such that $q\mathbb{Z}^n \subset \mathcal{L} \subset \mathbb{Z}^n$ for some $n, q > 0$ (these go under the name of q-ary lattice), or (ii) scale the fundamental region $\mathcal{P}$ and work using the natural discretization of points contained in $q\mathcal{P}$, which can all be represented using the coefficients in $\mathbb{Z}_q^n$ (which are integers) of the basis of the lattice that generates $\mathcal{P}$ (we will use this approach in the constructions that we present in Chapter 4).

2.1.1 Hard problems

We now introduce some lattice problems for which no efficient polynomial time algorithm is known, even by taking advantage of quantum computational power.

Shortest vector problem. The first problem deals with finding the shortest vector $\lambda_1(\mathcal{L})$ given a lattice $\mathcal{L}$.

Definition 12 (*Shortest vector problem* (SVP))

For any lattice $\mathcal{L}(B)$, the SVP *problem asks to find the shortest vector $\lambda_1(\mathcal{L}(B))$ of the lattice $\mathcal{L}(B)$.*

To be more precise, there are three variants of (exact) SVP, listed below:

- Search-SVP Given a lattice basis $B \in \mathbb{Z}^{n\times n}$, find $\boldsymbol{v} \in \mathcal{L}(B)$ such that $\|\boldsymbol{v}\| = \lambda_1(\mathcal{L}(B))$.
- Optimization-SVP: Given a lattice basis $B \in \mathbb{Z}^{n\times n}$, find the length $\lambda_1(\mathcal{L}(B))$.
- Decisional-SVP: Given a lattice basis $B \in \mathbb{Z}^{n\times n}$ and a real number $r \in \mathbb{R}$, determine whether $\lambda_1(\mathcal{L}(B)) \leq r$.

Notice that these problems have strong connections, e.g. by solving search-SVP one solves all the other problems, and by solving optimization-SVP one solves decisional-SVP as well.

To have an intuition of the hardness of the problem, see Figure 24, where a lattice is presented without plotting all the points.

On the other hand, by plotting all the points the solution become apparent (Figure 25). Notice that these problems are usually stated in terms of integer or rational values, so that

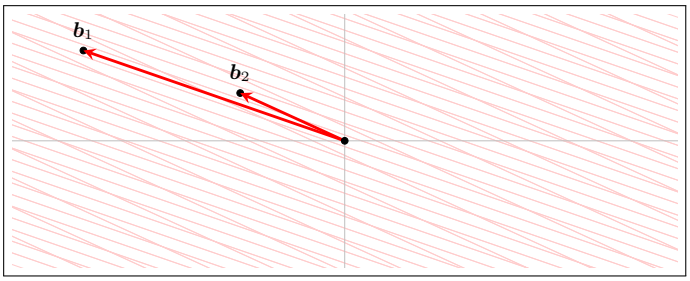


Figure 24: A sample of a SVP instance

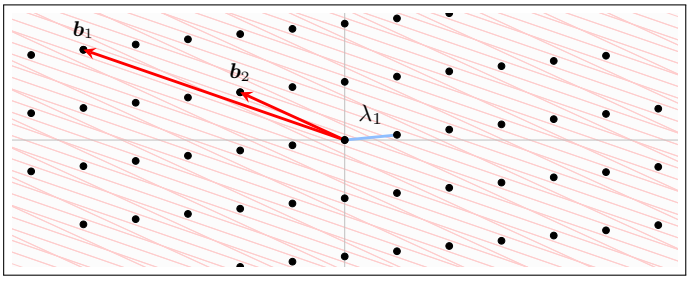


Figure 25: A sample of a SVP instance and the solution λ_1

the input can be represented in a finite number of bits and an instance can be treated as a standard computational problem. We will discuss more about this when dealing with q-ary lattices. Note that, if not specified, we will use the term SVP to refer to search-SVP. The problems can also be stated with respect to an approximation factor $\gamma > 1$.

- Search-SVP$_\gamma$: Given a lattice basis $B \in \mathbb{Z}^{m \times n}$, find $\boldsymbol{v} \in \mathcal{L}(B)$ such that $||\boldsymbol{v}|| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- Optimization-SVP$_\gamma$: Given a lattice basis $B \in \mathbb{Z}^{m \times n}$, find a real number $d \in \mathbb{R}$ such that $\lambda_1(\mathcal{L}(B)) \leq d \leq \gamma\lambda_1(\mathcal{L}(B)$.
- Decision-SVP$_\gamma$: Given a lattice basis $B \in \mathbb{Z}^{m \times n}$ and a real $r \in \mathbb{R}$, return 1 if $\lambda_1(\mathcal{L}(B)) \leq r$ and 0 if $\lambda_1 > \gamma \cdot r$. Otherwise, any value can be returned.

The decision version is a "gap" problem, and it is referred to as GapSVP$_\gamma$.

Definition 13 (GapSVP$_\gamma$)

For any $\mathcal{L}(B)$, $\gamma > 1$ and $r \in \mathbb{R}$, return 1 if $\lambda_1(\mathcal{L}) \leq r$ and 0 if $\lambda_1(\mathcal{L}) > \gamma \cdot r$.

This is a *promise* problem. Namely, there are inputs for which the algorithm behavior is unidentified (i.e., when $\lambda_1 \in [r, \gamma \cdot r)$). Nevertheless, given a GapSVP$_\gamma$ instance, we can assume that $\lambda_1(\mathcal{L}(B))$ lies in the "expected behavior" range.

We can also define another version of SVP, called γ-unique-SVP (γ-uSVP), useful to describe situations where the shortest vector is exceptionally short. As we will see, this is pretty common in cryptography, especially in homomorphic encryption.

Definition 14 (γ-uSVP)

For any lattice $\mathcal{L}$ such that $\lambda_2(\mathcal{L}) > \gamma\lambda_1(\mathcal{L})$ for some $\gamma > 1$, find $\boldsymbol{v} \in \mathcal{L}$ of length λ_1.

Notice that, as the approximation factor γ becomes larger, the shortest vector λ_1 becomes smaller, therefore the problem becomes easier Apparently, solving γ-uSVP seems easier than standard SVP [GN08]. We observe, though, that in cryptography, as we will see later, one does not always use the same problem, e.g., in some special lattices (algebraic modulo $X^n + 1$) solving decision-SVP, with an approximation factor of $\sqrt{n}$ is easy. In general, it is not.

[GN08]: Gama and P. Nguyen, *'Predicting Lattice Reduction'* . In: EUROCRYPT '08

The hardness of SVP has been topic of research for many years. One of the strongest results is due to Ajtai [Ajt98], who proved that Subset sum reduces to SVP under the ℓ_2 norm via a randomized reduction. Micciancio [Mic98] proved that SVP_γ is also NP-hard for small constant approximation factors $\gamma > 1$.

[Ajt98]: Ajtai, *'The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)'* . In: STOC '98

[Mic98]: Micciancio, *'The shortest vector in a lattice is hard to approximate to within some constant'* . In: FOCS '98

In general, the latest reductions are due to Bennet and Peikert [BP23], who gave a simpler proof of hardness of SVP under any $p \geq 1$ norm and constant approximation factor $\gamma < 2^{1/p}$. Notice that these proofs are still randomized, and it is a fascinating open problem to prove the hardness of SVP under a deterministic reduction. We refer the interested reader to a recent survey of Bennett [Ben23] for a nice presentation of the current state-of-the-art results and open problems regarding SVP.

[BP23]: Bennett and Peikert, *'Hardness of the (Approximate) Shortest Vector Problem: A Simple Proof via Reed-Solomon Codes'* . In: APPROX/RANDOM '23

[Ben23]: Bennett (2023), *'The Complexity of the Shortest Vector Problem'* . In: SIGACT News

Note that these results have a limited impact in cryptography, as in the latter we consider problems with polynomial approximation factors, which are not NP-hard [GG98; AR05] but lie in NP $\cap$ coNP. In particular, we can visualize the position of cryptography in the wide "approximation spectrum" in Figure 26. As it will be shown later, with an exponential approximation factor $\gamma = \Theta(2^n)$, we have an efficient algorithm that allows one to solve SVP_γ.

[GG98]: Goldreich and Goldwasser, *'On the limits of non-approximability of lattice problems'* . In: STOC '98

[AR05]: Aharonov and Regev (2005), *'Lattice problems in NP $\cap$ coNP'* . In: J. ACM

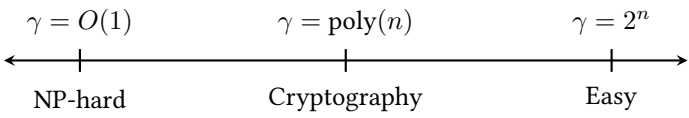


Figure 26: Hardness spectrum for SVP. Inspired by Figure 3 of [Ben23]

Closest vector problem. At this point, we introduce another well-known lattice problem, strongly related to decoding. It can be somehow seen as a formalization, in terms of computational complexity, of the distance function (2.9).

Definition 15 (*Closest vector problem* (CVP))

For any lattice $\mathcal{L}(B)$ and $\boldsymbol{t} \in \mathbb{R}^n$, called target vector, the CVP *problem asks to find the closest lattice point to the target $\boldsymbol{t}$. In other words find $\boldsymbol{v}$ such that:*

$$\arg\min_{\boldsymbol{v}\in\mathcal{L}} ||\boldsymbol{t} - \boldsymbol{v}||$$

The intuition behind this problem is rather simple: given a point lying in the span of the lattice, decode it to the closest lattice point. As before, we give a visual intuition of the hardness of the problem in Figure 27. Observe that, under a

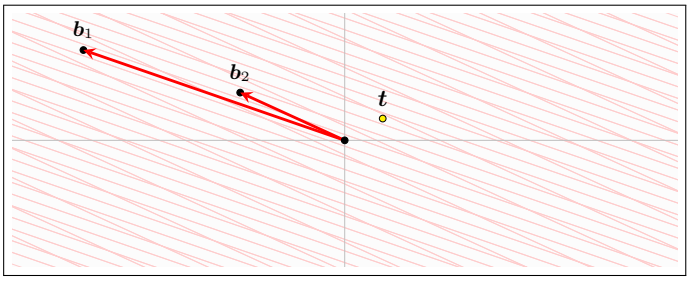


Figure 27: A sample of a CVP instance

long and non orthogonal basis, finding the closest vector of $\boldsymbol{t}$ is far from being a trivial task. Intuitively, a solution is to put a sphere centered in $\boldsymbol{t}$ and by letting it grow until it touches a lattice point (Figure 28).

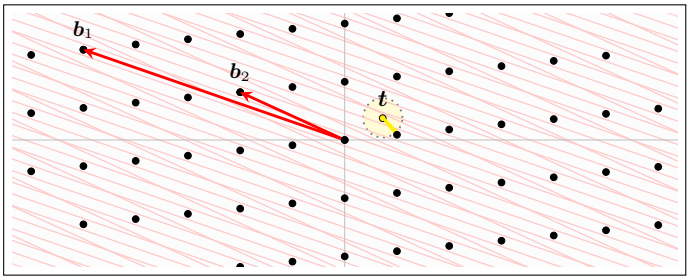


Figure 28: A sample of a CVP instance and the solution

Complexity results for CVP are somewhat similar to the ones for SVP – indeed, the hardness for SVP implies the same hardness for CVP [Gol+99]. Nevertheless, CVP behaves somehow better in terms of reductions, for example we can

[Gol+99]: Goldreich, Micciancio, Safra, and Seifert (1999), *'Approximating shortest lattice vectors is not harder than approximating closest lattice vectors'* . In: Information Processing Letters

find a rather simple proof of NP-hardness via a reduction to Subset sum [Mic01a].

[Mic01a]: Micciancio (2001), *'The hardness of the closest vector problem with preprocessing'* . In: IEEE Transactions on Information Theory

We conjecture that SVP might be slightly easier to solve rather than CVP, for instance Theorem 4.2 from [MG02] shows that $\text{CVP}_{\gamma'}$, with $\gamma'(n) > \sqrt{\frac{n}{(2/\gamma)^2-1}}$ can be reduced to $\mathcal{O}(n\log(n))$ calls to SVP_γ, with $\gamma \in [1,2)$. Nevertheless, it is still unclear whether CVP is NP-hard for such an approximation factor, therefore it is still unknown whether CVP is harder than SVP. We summarize the hardness results for both SVP and CVP in Table 1.

[MG02]: Micciancio and Goldwasser (2002), Complexity of Lattice Problems: a cryptographic perspective

Problem	Norm	Hardness	Ref
CVP_1	ℓ_p	NP-hard	[vEB81]
SVP_1	ℓ_∞	NP-hard	[vEB81]
SVP_1	ℓ_2	NP-hard*	[Ajt98]
$\text{SVP}_{\sqrt{2}}$	ℓ_2	NP-hard*	[Mic98]
SVP_c	ℓ_p	NP-hard*	[Kho04]
$\text{SVP}_{2(\log n)^{1/2-\epsilon}}$	ℓ_p	quasi-NP-hard	[Kho04]
$\text{SVP}_{2(\log n)^{1-\epsilon}}$	ℓ_p	quasi-NP-hard	[Kho04]

Table 1: History of hardness for standard lattice problems. The * symbol means under randomized reductions. Quasi-NP-hard means that the reduction runs in super-polynomial, though still sub-exponential. Taken from a 2015 lecture by Vaikuntanathan [Vai15]

Similarly to the case of SVP and uSVP, we can define a particular instance of CVP in which the distance from the target $\boldsymbol{t}$ to the closest lattice point is promised to be exceptionally small.

Definition 16 (α-*Bounded distance decoding* (BDD_α))

For any lattice $\mathcal{L}(B)$ and $\boldsymbol{t} \in \mathbb{R}^n$, the BDD_α problem asks to find the closest lattice point to the target $\boldsymbol{t}$ when the distance $\mathcal{L} - \boldsymbol{t}$ is at most $\alpha\lambda_1(\mathcal{L}(B))$, with $\alpha \leq \frac{1}{2}$.

Throughout the thesis we will ignore the particular promise factor α and simply talk about a BDD instance $(\boldsymbol{t}, \mathcal{L})$ whenever $\text{dist}(\boldsymbol{t}, \mathcal{L}) < \frac{1}{2}\lambda_1(\mathcal{L})$. In geometrical terms , we can see CVP as the problem of decoding a point inside some sphere of radius at most the covering radius, while BDD as the problem of decoding a point within a radius equal to the packing radius. This, however, is a slight abuse of notation since CVP does not have a "distance constraint", whereas BDD does.

An important result by Lyubashevsky and Micciancio [LM09] proves the equivalence between BDD, GapSVP and uSVP, up some small approximation factor $\sqrt{n/\log(n)}$.

[LM09]: Lyubashevsky and Micciancio, *'On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem'* . In: CRYPTO '09

2.1.2 The LLL algorithm

One of the most celebrated algorithms in lattice theory is the Lenstra-Lenstra-Lovasz (LLL) algorithm [LLL82]. At first, it was introduced as a solution to the problem of finding the decomposition of a polynomial $p(X) \in \mathbb{Q}[X]$ (or in $\mathbb{Z}[X]$) into irreducible factors. It turns out that this algorithm has many applications in Computer science, and in particular can be used to solve SVP with $\gamma = (2/\sqrt{3})^n$ approximation factor, with n being the lattice dimension.

[LLL82]: A. K. Lenstra, H. W. Lenstra, and Lovász (1982), *'Factoring polynomials with rational coefficients'* . In: Mathematische Annalen

From a certain perspective, LLL can be seen as a generalization of the Lagrange-Gauss algorithm that allows one to find an "optimal basis", in terms of orthogonality and shortness, in two dimensions. The latter case is optimal, in the sense that a Lagrange-Gauss-reduced basis contains the shortest vector λ_1. If the reader is not familiar Lagrange-Gauss algorithm, we suggest Section 17.1 of [Gal12] or Chapter 6 of [vW23], which give a somewhat soft introduction to lattice basis reduction, since in two dimensions the process can also be visualized.

[Gal12]: Galbraith (2012), Mathematics of Public Key Cryptography

[vW23]: van Woerden (2023), *'Lattice cryptography: from cryptanalysis to New Foundations'*, *Ph.D. Thesis, Leiden University.*

For our needs, we will only need to apply LLL over full-rank lattice bases, so we will only describe that case, although it can be generalized to any basis.

Definition 17 (δ-LLL-*reduced basis*)

A lattice basis $B \in \mathrm{GL}_n(\mathbb{R})$ is δ-LLL-reduced *if the two following conditions hold:*

- *(*Size reduction*) For each $0 < j < i \leq n$, it holds that*

$$|\mu_{i,j}| \leq \frac{1}{2},$$

  where $\mu_{i,j} := \frac{\langle \boldsymbol{b}_i, \tilde{\boldsymbol{b}}_j \rangle}{\langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle}$ is the Gram-Schmidt coefficient.
- *(*Lovasz condition*) For any pair of consecutive vectors $\boldsymbol{b}_i, \boldsymbol{b}_{i+1}$:*

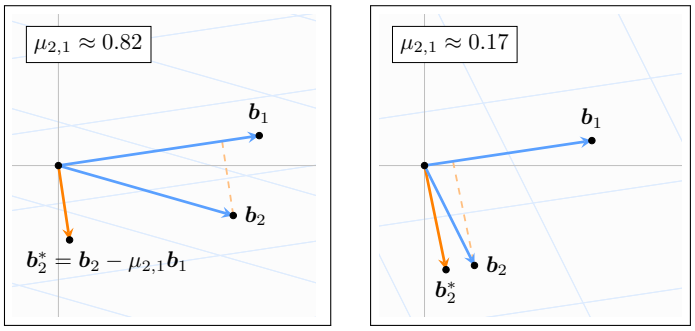
$$\delta \|\tilde{\boldsymbol{b}}_i\|^2 \leq \|\tilde{\boldsymbol{b}}_{i+1}\|^2 + \|\mu_{i+1,i} \cdot \tilde{\boldsymbol{b}}_i\|^2$$

  with $\frac{1}{4} < \delta \leq 1$.

Note that, if omitted, we assume $\delta = 3/4$, as used in the original proofs in [LLL82]. For larger values, the length of the basis can be reduced even more, but the runtime is much worse in practice. Moreover, we can not show that for $\delta = 1$ the algorithm terminates in polynomial time. The reduced basis definition can be interpreted in geometrical terms. In

particular, the first *size reduction* condition is required to check the orthogonality of the basis[8]. For every pair of vectors $\boldsymbol{b}_i, \boldsymbol{b}_j$, we check their Gram-Schmidt coefficient. It is possible to visualize such quantity by (i) computing an orthogonal projection of one vector $\boldsymbol{b}_i$ to another $\boldsymbol{b}_j$, and (ii) to check "how much of $\boldsymbol{b}_i$ length is covered by the projection of $\boldsymbol{b}_j$". In Figure 29 we show an example, in particular **(a)** shows a case where the μ coefficient is larger than 0.5, while **(b)** a case where the coefficient is smaller.

8: A simple inner product could be used to check orthogonality, but the Gram-Schmidt coefficient captures the orthogonal component of some vector with respect to another. So it can be treated as an "quantity of orthogonality".

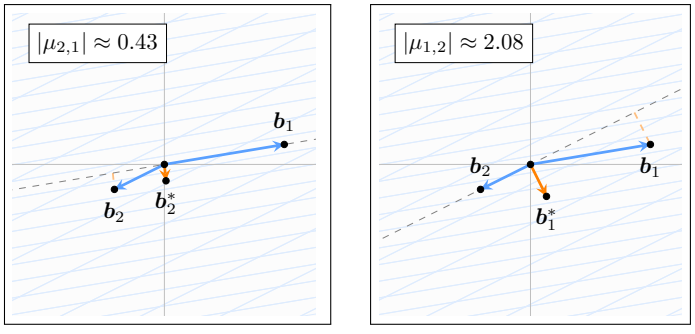


(a) Non orthogonal vectors

(b) Orthogonal vectors

Figure 29: Size reduction checks that the Gram-Schmidt coefficient between pairs of vectors is smaller than $\frac{1}{2}$, this is a necessary condition for orthogonality

We can also see that an interpretation is how much of $\boldsymbol{b}_1$ is covered by an orthogonal projection of $\boldsymbol{b}_2$ onto $\boldsymbol{b}_1$ – approximately 82 percent of its length in the first case. Notice that, although necessary, this condition is not sufficient to check orthogonality between two vectors. Indeed, consider a case where the angle between two vectors is approximately $180°$. In such case, the Gram-Schmidt coefficient might be smaller than $1/2$, although the two vectors are far from being orthogonal, refer to Figure 30 **(a)**.



(a) False positive: satisfying $|\mu_{2,1}| < 0.5$

(b) True positive: not satisfying $|\mu_{2,1}| < 0.5$

Figure 30: Swapping the ordering of the Gram-Schmidt from $\mu_{2,1}$ to $\mu_{1,2}$ allows to find "false positives" for the orthogonality check

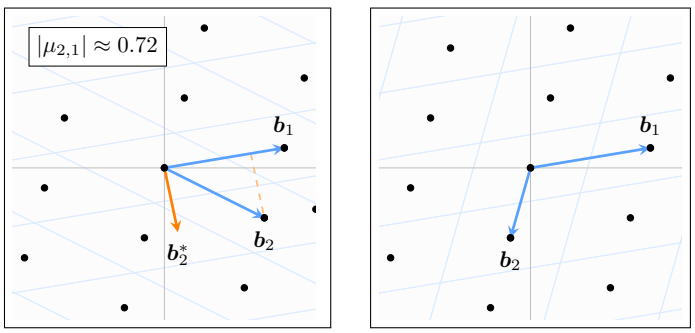
The Lovasz condition is precisely required for this reason – if the check fails, it is possible to swap the order of the vectors and re-compute the coefficients, as in Figure 30 **(b)**.

We saw that a δ-LLL-reduced basis satisfies Definition 17, and we anticipated that there is a polynomial time algorithm that given any basis $B \in \mathrm{GL}_n(\mathbb{R})$ finds the corresponding δ-LLL-reduced basis. Before presenting the algorithm, we introduce its main subroutine, namely SizeReduce (Algorithm 1).

```
1 function SizeReduce (B)
2 for i := 2 to n do
3     for j := (i − 1) downto 1 do
4         μ_{i,j} = ⌊⟨b_i, b̃_j⟩ / ⟨b̃_j, b̃_j⟩⌉
5         b_i := b_i − μ_{i,j} · b_j
6 return B
```

Algorithm 1: Size reduction procedure

The intuition behind this phase is about removing multiples of $\boldsymbol{b}_j$ from $\boldsymbol{b}_i$. Notice that this creates a shorter basis – as we are reducing the norm of $\boldsymbol{b}_i$ – but at the same time we are not changing the lattice as we are subtracting a vector (scaled by an integer) that is already a lattice point, see as a reference Figure 31.



(a) Before size-reduction procedure

(b) After size-reduction procedure

Figure 31: Reducing the size of a basis starting from $\boldsymbol{b}_1, \boldsymbol{b}_2$ such that $\lfloor \mu_{2,1} \rceil = 1$. The new vector $\boldsymbol{b}_2$ becomes $\boldsymbol{b}_2 := \boldsymbol{b}_2 - 1 \cdot \boldsymbol{b}_1$

The LLL algorithm repeats this phase as it allows to progressively reduce the value of Gram-Schmidt coefficients $\mu_{i,j}$. If some $|\mu_{i,j}| < \frac{1}{2}$, but the vectors are far from being orthogonal, they are swapped – following Lovasz condition. This is repeated until the basis is not δ-LLL-reduced.

Observe that the Lovasz condition can also be written as

$$\|\tilde{\boldsymbol{b}}_{i+1}\|^2 \geq (\delta - \mu_{i+1,i}^2) \cdot \|\tilde{\boldsymbol{b}}_i|^2. \tag{2.11}$$

Considering that $\mu_{i+1,i}$ captures the information about the angle between $\boldsymbol{b}_i$ and $\boldsymbol{b}_{i+1}$, this condition ensures that the orthogonalized vector is not significantly larger than the subsequent one, with respect to $\mu_{i+1,i}$ and the δ we are picking[9]. Larger values of δ make the condition harder to verify, causing the algorithm to require more time to produce the reduced basis – but at the same time obtaining a basis of a higher "quality".

9: This is what happens, for instance, in Figure 30, where the norm $\|\tilde{\boldsymbol{b}}_2\|$ is very small with respect to $\|\tilde{\boldsymbol{b}}_1\|$, weighted by $\delta - (\mu_{2,1})^2 \approx 0.57$, assuming $\delta = 3/4$

Hence, for an LLL-reduced basis, even if GSO vectors $\tilde{\boldsymbol{b}}_i$ can become shorter, their length can not decrease too quickly. We present a pseudocode of LLL in Algorithm 2.

```
1 function LLL (B, δ)
2     B := SizeReduce(B)
3     if δ‖b_i‖² > ‖b_{i+1}‖² + ‖μ_{i+1,i} · b_i‖² for some i then
4         Swap b_i and b_{i+1}
5         return LLL(B, δ)
6     return B
```

Algorithm 2: The LLL algorithm

The just presented algorithm is correct and terminates in polynomial time as stated by the following lemma.

Lemma 3 ([LLL82])

For any integer basis $B \in GL_n(\mathbb{Z})$ and $\delta \in (\frac{1}{4}, 1)$, Algorithm X is correct and has complexity

$$O(n^2 \log(\max\|\boldsymbol{b}_i\|)).$$

We refer the interested reader to the lecture notes of Micciancio [Mic12] for a detailed proof of this lemma.

[Mic12]: Micciancio (2012), Lecture 3: The LLL Algorithm, *Course Lecture Notes for CSE 206A: Lattice Algorithms and Applications, UCSD* (online)

An LLL-reduced basis has the following properties that directly follow from the definition.

Corollary 2 (*Bounding the first LLL-reduced vector*)

For any δ-LLL-reduced basis, it holds that:

$$\begin{aligned} \|\boldsymbol{b}_1\| &\leq (\delta - \frac{1}{4})^{-(n-1)/2} \cdot \lambda_1(\mathcal{L}), \\ \|\boldsymbol{b}_1\| &\leq (\delta - \frac{1}{4})^{-(n-1)/4} \cdot \mathrm{vol}(\mathcal{L})^{1/n}. \end{aligned} \tag{2.12}$$

Proof. Let $\gamma = \sqrt{1/(\delta - \frac{1}{4})}$. By combining both LLL conditions we obtain:

$$\|\boldsymbol{b}_1\| = \|\tilde{\boldsymbol{b}}_1\| \leq \gamma \cdot \|\tilde{\boldsymbol{b}}_2\| \leq \cdots \leq \gamma^{n-1} \cdot \|\tilde{\boldsymbol{b}}_n\|.$$

In general, we have that $\|\boldsymbol{b}_1\| \leq \gamma^{n-1} \min_i \|\tilde{\boldsymbol{b}}_i\|$, since the norm of the GSO vectors is nonincreasing (by Lovasz condition). Notice that $\min_i \|\tilde{\boldsymbol{b}}_i\| \leq \lambda_1(\mathcal{L})$ by Lemma 2, this leads to:

$$\|\boldsymbol{b}_1\| \leq \gamma^{n-1}\lambda_1 = (\delta - \frac{1}{4})^{-(n-1)/2} \cdot \lambda_1(\mathcal{L}).$$

The second observation comes from the fact that, by Definition 7, we have:

$$\mathrm{vol}(\mathcal{L})^{1/n} = \left(\prod_i^n \|\tilde{\boldsymbol{b}}_i\|^{1/n}\right) \geq$$
$$\left(\prod_i^n \|\tilde{\boldsymbol{b}}_1\| \cdot \gamma^i\right)^{1/n} = \|\tilde{\boldsymbol{b}}_1\| \cdot \gamma^{(n-1)/2}.$$

□

Notice that, in the unpractical case of $\delta = 1$, LLL can be seen as an algorithm to find a vector within the Minkoswki bound (Theorem 1)

Corollary 3 (*Bounding the GSO vectors for LLL-reduced basis*)

For any δ-LLL-reduced basis, it holds that

$$\|\tilde{\boldsymbol{b}}_{i+1}\| \geq \sqrt{\delta - \frac{1}{4}}\|\tilde{\boldsymbol{b}}_i\|,$$

for $i = 1, \ldots n-1$.

Proof. Given the Lovasz condition:

$$\delta\|\tilde{\boldsymbol{b}}_i\|^2 \leq \|\tilde{\boldsymbol{b}}_{i+1}\|^2 + \|\mu_{i+1,i} \cdot \tilde{\boldsymbol{b}}_i\|^2,$$

that can be rewritten equivalently as

$$\|\tilde{\boldsymbol{b}}_{i+1}\|^2 \geq (\delta - \mu_{i+1,i}^2)\|\tilde{\boldsymbol{b}}_i\|^2,$$

we plug in the size reduction condition, i.e., $|\mu_{i+1,i}| < \frac{1}{2}$, to obtain

$$\|\tilde{\boldsymbol{b}}_{i+1}\|^2 \geq (\delta - \frac{1}{4})\|\tilde{\boldsymbol{b}}_i\|^2.$$

The result follows by taking square roots. □

We refer the interested reader to [NV09] for a broad and historical survey on LLL and to [DPS25] for the latest implementation of the algorithm.

[NV09]: P. Q. Nguyen and Valle (2009), The LLL Algorithm: Survey and Applications

[DPS25]: Ducas, Pulles, and M. Stevens (2025), Towards a Modern LLL Implementation (Cryptology ePrint Archive, Paper 2025/774)

[Bab86]: Babai (1986), *'On Lovász' lattice reduction and the nearest lattice point problem'* . In: Combinatorica

2.1.3 Babai nearest plane algorithm

One of the most important algorithms in lattice theory is due to Babai [Bab86], and it allows to solve the CVP problem within an exponential approximation factor. This algorithm has a very intuitive geometrical intuition from Babai's fundamental region.

Definition 18 (*Babai's fundamental region*)

For any lattice basis $B \in \mathbb{R}^n$ and its Gram-Schmidt orthogonalization $\tilde{B}$, the Babai's fundamental region *is:*

$$\mathcal{P}(\tilde{B}) = \tilde{B} \cdot \left[-\frac{1}{2}, \frac{1}{2}\right)^n .$$

Consider the lattice in Figure 32, and its corresponding Babai's fundamental region (in orange). Informally, the goal of Babai's

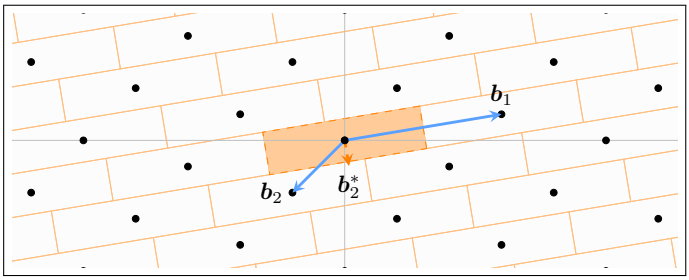


Figure 32: A lattice and its Babai's fundamental region

nearest plane algorithm is to round any target point inside any "orange block" to the corresponding lattice point. Since Babai's fundamental region is a (n-dimensional) rectangle, it is easy to decode any point inside it using projections. We will better discuss the "quality" of a basis in Section 2.1.4, although notice that with a quasi-orthogonal basis we can decode points by simply rounding the point to the rectangle in which it lies (Figure 33). On the other hand, usually a bad basis consists of mixed long and short vectors, far from being orthogonal. The result, in terms of "tiling", is very bad as decoding using the same approach fails (Figure 34).

We now describe the algorithm and we will present some results that can define in which cases CVP can be solved efficiently (Algorithm 3).

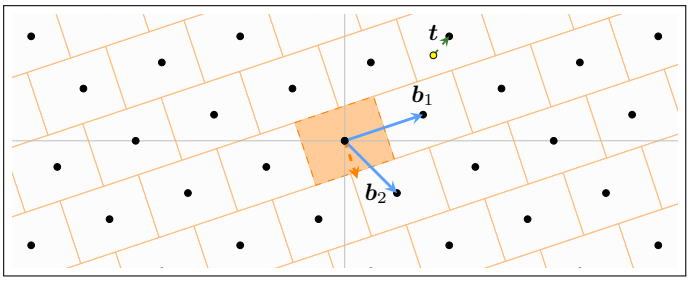


Figure 33: A lattice generated by a good basis and its Babai's fundamental region

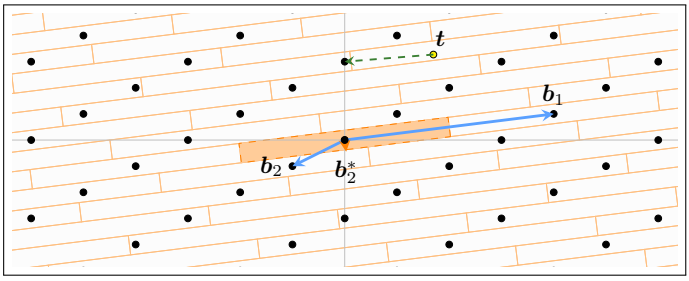


Figure 34: A lattice generated by a bad basis and its Babai's fundamental region

function BabaiNearestPlane $(B, \boldsymbol{t})$
  $B := \text{LLL}(B, \delta = 3/4)$
  $\boldsymbol{p} \leftarrow \boldsymbol{t}$
  $\boldsymbol{e} := \boldsymbol{0}$
  for $i := n$ downto 1 **do**
    $k_i := \lfloor \frac{\langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_i \rangle}{\langle \tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_i \rangle} \rceil$
    $\boldsymbol{p} := \boldsymbol{p} - k_i \boldsymbol{b}_i$
    $\boldsymbol{c} := \boldsymbol{c} + k_i \boldsymbol{b}_i$
return $\boldsymbol{p}, \boldsymbol{e}$

Algorithm 3: Babai nearest plane algorithm

We can have an intuition on the algorithm as it rounds some target to the unique point in the fundamental region $\mathcal{P}(\tilde{B})$.

Lemma 4 (*Babai's nearest plane correctness*)

For any lattice $\mathcal{L}(B)$ and Babai's fundamental region $\mathcal{P}(\tilde{B})$, let $\boldsymbol{t} \in span(\mathcal{L})$ be any target and $\boldsymbol{p}, \boldsymbol{c} := BabaiNearestPlane(B, \boldsymbol{t})$. Then $\boldsymbol{p} := \boldsymbol{t} - \boldsymbol{e} \in \mathcal{P}(\tilde{B})$, namely Babai's nearest plane algorithms computes the unique close vector $\boldsymbol{c}$ with error $\boldsymbol{e}$ in $\mathcal{P}(\tilde{B})$.

At each iteration, the target $\boldsymbol{t} \in \text{span}(\mathcal{L})$ is projected onto the hyperplane defined by the i-th Gram-Schmidt vector. The "nearest plane" choice is given by the rounding when computing k_i, the longer $\tilde{\boldsymbol{b}}_i$, the more accurate the rounding. Consider the denominator $\langle \tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_i \rangle$: if it is large, the impact of

the "error" naturally present in the numerator, will be smaller, meaning that the rounding becomes more stable. On the other hand, short Gram-Schmidt vectors cause the denominator to be very small, increasing the impact of the error in the numerator. In such a case the rounding is somehow fragile and leads to large "swings" during the rounding procedure.

We formalize this intuitions in the following.

Corollary 4 (*Babai's nearest plane solves CVP$_\gamma$* [Bab86])

For any $\delta \in [\frac{1}{4}, 1)$, Babai's nearest plane algorithm with a δ-LLL-reduced basis allows to solve CVP_γ, with $\gamma < \sqrt{n}(\delta - \frac{1}{4})^{-n/2}$. In particular there exists an efficient algorihtm that solves CVP_γ for $\gamma = 2^{O(n)}$.

Also, we can bound the norm of the "error" $\boldsymbol{e}$ given as output from BabaiNearestPlane.

Corollary 5 (*Bound on BabaiNearestPlane error*)

For any lattice $\mathcal{L}(B)$ and target $\boldsymbol{t} \in span(\mathcal{L}(B))$, let $(\boldsymbol{p}, \boldsymbol{e}) := BabaiNearestPlane(B, \boldsymbol{t})$. Then

$$\|\boldsymbol{t} - \boldsymbol{c}\|^2 = \|\boldsymbol{e}\|^2 \leq \frac{1}{4}\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2 \leq \frac{1}{4}\sum_{i=1}^{n} \|\boldsymbol{b}_i\|^2.$$

Moreover, we can find the unique closest lattice vector to $\boldsymbol{t}$, thus solving exact CVP, if $\text{dist}(\mathcal{L}, \boldsymbol{t}) \leq \frac{1}{2}\min_i \|\tilde{\boldsymbol{b}}_i\|$.

2.1.4 Basis quality and profiles

Until now, we sometimes used the term *good basis* or *bad basis* without actually defining what it means. Remark that every lattice $\mathcal{L}$ admits an infinite number of bases, since the number of unimodular matrices $U \in \text{GL}_n(\mathbb{Z})$ is infinite (Lemma 1). Nevertheless, some bases (short and orthogonal) allows to solve some hard problems more efficiently. Think of, as an example, the identity basis $B = I_n$ for $\mathbb{Z}^n$. Solving CVP there is as simple as rounding the coordinates $\boldsymbol{x} \in \mathbb{R}^n$ of the target $\boldsymbol{t} = B\boldsymbol{x} \in \mathbb{R}^n$. A quantity relative to the "quality" of a basis is the *orthogonality defect*.

Definition 19 (*Orthogonality defect*)

The orthogonality defect *of a lattice basis B is:*

$$\delta(B) = \frac{\prod_{i=1}^{n} \|\boldsymbol{b}_i\|}{\prod_{i=1}^{n} i\|\tilde{\boldsymbol{b}}_i\|} \geq 1.$$

Notice that with full-rank bases it holds that $\prod_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\| = \det(B)$, so $\delta(B) = \prod_{i=1}^{n} \|\boldsymbol{b}_i\|/\det(B)$. A basis is good if and only if $\delta(B)$ is close to 1; the best case is when the norms of the basis vectors and the GSO vectors are equal – this implies $\delta(B) = 1$ and perfect orthogonality. This quantity also captures the implicit connection between the length and the orthogonality of the basis vectors, via the Gram-Schmidt vectors.

In order to efficiently solve hard problems over lattices we need to minimize such defect, although the best we can do efficiently is to compute an LLL reduction (or other more advanced basis reductions, which are out of scope for this thesis).

Babai's nearest plane algorithm represents a solution to decode any target $\boldsymbol{t} \in \text{span}(\mathcal{L})$ to the closest $\boldsymbol{v} \in \mathcal{L}$ such that $\boldsymbol{t} - \boldsymbol{v} \in \mathcal{P}(\tilde{B})$, so the role of the GSO vectors is crucial in cryptanalysis. In particular, to build secure schemes – or problems hard to solve – we want to build a basis such that $\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2$ is very large, fixed some $\prod_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|$

Consider, without loss of generality, some basis for $\mathbb{Z}^n$ such that its determinant is 1. Given that the determinant is an invariant property of the lattice (i.e., $\det(\mathcal{L}(B)) = \text{vol}(\mathcal{P}(B))$), we have that $\sum_{i}^{n} \|\tilde{\boldsymbol{b}}_i\|^2$ is maximized when

$$\|\boldsymbol{b}_1\| = \|\boldsymbol{b}_2\| = \cdots = \|\boldsymbol{b}_n\| = \text{vol}(\mathcal{P})^{1/n},$$

by Cauchy-Schwartz's inequality $|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| \leq \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|$. Indeed, a perfectly orthogonal basis with equal length GSO vectors has the most powerful "decoding power", and an orthogonality defect $\delta(B) = 1$.

On the other hand, by changing the basis of the considered lattice – e.g., via a unimodular transformation – we might end up with an extremely large norm for some vector $\|\tilde{\boldsymbol{b}}_i\|$, which implies that some other GSO vector must be tiny to keep the determinant constant. Imbalance between GSO vectors implies poor decoding performance.

A non orthogonal basis is defined by (i) a large first GSO norm $\|\tilde{\boldsymbol{b}}_1\|$, which in turn implies a large amount of $\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2 \geq$

$\|\tilde{\boldsymbol{b}}_1\|$, and (ii) a very small last GSO norm $\|\boldsymbol{b}_n\|$, which implies poor decoding performance. To get another intuition, think of Babai's decoding algorithm, which starts from the last $\tilde{\boldsymbol{b}}_n$ and ends at $\tilde{\boldsymbol{b}}_1$. If the last GSO vector is small, the first rounding will be inaccurate, even a small error will force the algorithm to pick a very far lattice point, since the decoding phase involves a tiny parallelepiped section. When the last iteration is reached, the "best" vector (in term of length) $\tilde{\boldsymbol{b}}_1$ will not be useful as the decoding process will already be "ruined".

A convenient tool that allows one to quickly analyze the quality of a basis is the *profile*.

Definition 20 (*Profile of a basis*)

The profile *of a lattice basis $B \in GL_n(\mathbb{R})$ is a n-tuple consisting of the Gram-Schmidt norms:*

$$(\|\tilde{\boldsymbol{b}}_1\|, \|\tilde{\boldsymbol{b}}_2\|, \ldots, \|\tilde{\boldsymbol{b}}_n\|).$$

We can plot the log-profile (i.e., with a logarithmic y-axis) of some basis to have an idea about its quality. The profile of a good basis will be approximately flat and slowly descending, indicating that the norms of the GSO vectors are close. On the other hand, the profile of a bad basis is steep as it starts from a very large $\|\tilde{\boldsymbol{b}}_1\|$ and quickly decreases towards a small value.

Notice that the profile of an LLL-reduced basis does not decrease too steeply, because of Lovasz condition (Eq. 2.11) and, in turn, because of Lemma 3.

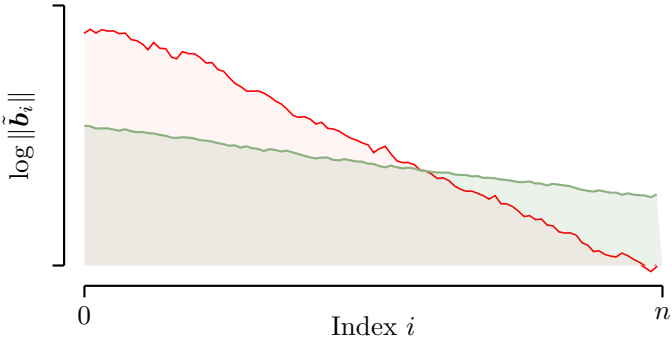


Figure 35: Example of two profiles for a good basis (green) and a bad basis (red)

The spectrum of basis reduction techniques go from the polynomial LLL to the exponential time HKZ, with BKZ algorithm in the middle [KZ73]. We refer to [Sch87].

[Sch87]: Schnorr (1987), *'A hierarchy of polynomial time lattice basis reduction algorithms'* . In: Theoretical Computer Science

2.1.5 Duality

Given some lattice $\mathcal{L}$, it is always possible to define its *dual* $\mathcal{L}^*$ as follows.

Definition 21 (*Dual lattice*)

For any (primal) lattice $\mathcal{L}$, the dual lattice $\mathcal{L}^*$ *is the set of vectors that have integer inner product with all the points in $\mathcal{L}$. Formally:*

$$\mathcal{L}^* := \{\boldsymbol{x} \in \text{span}(\mathcal{L}) : \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}, \forall \boldsymbol{y} \in \mathcal{L}\}. \tag{2.13}$$

Notice that the primal and the dual lattice do not usually share the same points, i.e., one does not contain the other.

Theorem 3 (*Basis for the dual*)

The dual of $\mathcal{L}(B)$ is $\mathcal{L}(B)^* = \mathcal{L}(B(B^tB)^{-1})$.

Proof. We prove this with two different arguments. (i) Every integer combination of columns of B^* lies in $\mathcal{L}^*$: a point $\boldsymbol{v} \in \mathbb{R}^n$ is in the dual lattice if and only if it has integer inner product with all the lattice points in the primal. Hence, take $\boldsymbol{w} \in \mathbb{Z}^n$ and set $\boldsymbol{y} = B(B^tB)^{-1}\boldsymbol{w}$. Then

$$B^t\boldsymbol{y} = B^tB(B^tB)^{-1}\boldsymbol{w} = \boldsymbol{w} \in \mathbb{Z}^n$$

so $\boldsymbol{y} \in \mathcal{L}^*$, and in general $\{B(B^tB)^{-1}\boldsymbol{w} : \boldsymbol{w} \in \mathbb{Z}^n\} \subseteq \mathcal{L}^*$. Next, (ii) every vector of $\mathcal{L}^*$ is an integer combination of the columns of B^*: let $\boldsymbol{y} \in \mathcal{L}^*$. Since $\boldsymbol{y} \in \text{span}(B)$, we can write $\boldsymbol{y} = B\boldsymbol{x}$ for some $\boldsymbol{x} \in \mathbb{R}^n$. The duality condition gives

$$B^t\boldsymbol{y} = B^tB\boldsymbol{x} \in \mathbb{Z}^n.$$

Notice that B^tB is invertible and in $\mathbb{Z}^{n\times n}$, thus we have that $\boldsymbol{x} = (B^tB)^{-1}B^tB\boldsymbol{x}$, therefore

$$\boldsymbol{y} = B\boldsymbol{x} = B(B^tB)^{-1}\underbrace{B^tB\boldsymbol{x}}_{\in\mathbb{Z}^n},$$

so $\boldsymbol{y}$ is an integer combinations of the columns of $B(B^tB)^{-1}$, therefore $\mathcal{L}^* \subseteq \{B(B^tB)^{-1}\boldsymbol{w} : \boldsymbol{w} \in \mathbb{Z}^n\}$. By putting together (i) and (ii), the result follows. □

Moveover, when considering full-rank bases, we have the following simpler result.

Corollary 6 (*Basis for dual, when full-rank*)

The dual of $\mathcal{L}(B)$, when $B \in GL_n(\mathbb{R})$ is full-rank, is $\mathcal{L}(B^{-t})$.

Proof. In this special case the proof is simple. A point $\boldsymbol{v} \in \mathbb{R}^n$ is in the dual lattice if and only if it has integer inner product with all the lattice points in the primal, when B is full-rank we have $B^t\boldsymbol{v} \in \mathbb{Z}^n$ (remark that the basis vectors of $\mathcal{L}(B)$ are all the columns of B). Equivalently, we can write down this condition as $\boldsymbol{v} \in B^{-t}\mathbb{Z}^n$, therefore B^{-t} is a basis for the dual lattice. □

We can also obtain this result intuitively from the definition of a dual lattice: given some $B \in \mathrm{GL}_n(\mathbb{Z})$, then $\boldsymbol{v} \in \mathbb{R}^n$ is a dual vector if and only if $B^t\boldsymbol{v} \in \mathbb{Z}^n$, or equivalently $\boldsymbol{v} \in B^{-t}\mathbb{Z}^n = \mathcal{L}(B^{-t})$.

Lemma 5 (*Symmetry of duality*)
The dual of $\mathcal{L}^*$ is $(\mathcal{L}^*)^* = \mathcal{L}$.

Proof. We know that the basis of the dual of $\mathcal{L}(B)$ is $B(B^tB)^{-1}$. By fixing $C = B(B^tB)^{-1}$, the basis of $(\mathcal{L}^*(B))^*$ is:

$$C(C^tC)^{-1} = B(B^tB)^{-1}(C^tC)^{-1}$$

Notice that

$$(C^tC)^{-1} = (B(B^tB)^{-t} \cdot B(B^tB)^{-1})^{-1} = B^tB$$

Leading to

$$C(C^tC)^{-1} = B(B^tB)^{-1} \cdot (C^tC)^{-1} = B$$

□

Lemma 6 (*Determinant of the dual*)
For any lattice $\mathcal{L}$, $det(\mathcal{L}^*) = (det(\mathcal{L}))^{-1}$.

Proof. Given the basis B for the primal lattice and $C = B(B^tB)^{-1}$ for the dual, simply consider $\det(\mathcal{L}^*) = \sqrt{\det(C^tC)}$, and derive

$$\begin{aligned}
\sqrt{\det(C^tC)} &= \sqrt{\det((B(B^tB)^{-1})^t \cdot B(B^tB)^{-1})} \\
&= \sqrt{\det((B^tB)^{-1})^t \cdot B^t \cdot B(B^tB)^{-1})} \\
&= \sqrt{\det((B^tB)^{-t})} \\
&= 1/\sqrt{\det(B^tB)} = 1/\det(\mathcal{L})
\end{aligned}$$

□

Among many useful usages of dual lattices, we will be very much interested in the fact that any vector in $\mathcal{L}$ can be used to partition $\mathcal{L}^*$ into different layers, where each layer ℓ is

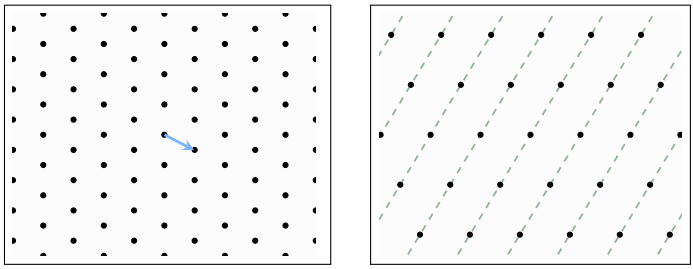
composed of the points $\boldsymbol{v} \in \mathcal{L}^*$ such that $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \ell$, for some primal point $\boldsymbol{w} \in \mathcal{L}$.

Definition 22 (*Dual partitions*)

For any lattice $\mathcal{L}(B)$, its dual $\mathcal{L}(B)^*$, and a lattice point $\boldsymbol{v} \in \mathcal{L}$, we can define a set of partitions L_i, *where each*

$$L_i(\boldsymbol{v}) = \{\boldsymbol{w} \in \mathcal{L}^* : \langle \boldsymbol{v}, \boldsymbol{w} \rangle = i\}.$$

Consider the example in Figure 36 and notice that by picking some vector in $\mathcal{L}$, we create partitions in $\mathcal{L}$ orthogonal to $\boldsymbol{v}$. Of



(a) Primal lattice $\mathcal{L}$ **(b)** Dual lattice $\mathcal{L}^*$

Figure 36: Primal and dual lattice, each dashed line in **(a)** refers to a partition ℓ of points such that each point has inner product ℓ with the picked $\boldsymbol{v} \in \mathcal{L}^*$

course, by symmetry, the same holds the other way around.

Lemma 7 (*Distance between partitions*)

For any $\boldsymbol{v} \in \mathcal{L}^*$ and $i \neq j \in \mathbb{Z}$, the distance between two partitions $L_i(\boldsymbol{v}), L_j(\boldsymbol{v})$ in $\mathcal{L}$ is $|i - j|/\|\boldsymbol{v}\|$.

Proof. Consider two layers $L_i(\boldsymbol{v})$ and $L_j(\boldsymbol{v})$. The distance between two hyperplanes defined by $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = i$ and $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = j$ is $|i - j|$ unless $\boldsymbol{v}$ has unit length. Since this does not hold in general, we "normalize" the hyperplanes as $\langle \hat{\boldsymbol{v}}, \boldsymbol{w} \rangle = i$ and $\langle \hat{\boldsymbol{v}}, \boldsymbol{w} \rangle = j$, where $\hat{\boldsymbol{v}} = \boldsymbol{v}/\|\boldsymbol{v}\|$, so that the distance is $i/\|\boldsymbol{v}\| - j/\|\boldsymbol{v}\|$. □

Hence, the width of such dual partitions is given by $1/\|\boldsymbol{v}\|$, with $\boldsymbol{v} \in \mathcal{L}$ being the picked vector in the primal lattice. This immediately suggests that the "best" partitions, in terms of width, are given by the shortest vector in the primal lattice.

From target to syndrome. The dual lattice can give a different perspective to the CVP problem. In particular, if we are given without loss of generality some target $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e} \in \text{span}(\mathcal{L}(B)^*)$, where $\boldsymbol{v} \in \mathcal{L}^*$ and $\boldsymbol{e} \in \mathbb{R}^n$ is an "error", we can define the *syndrome* of $\boldsymbol{t}$ as follows. We know that $\mathcal{L}(B)$

contains all the points that have integer inner product with the points of $\mathcal{L}(B)^*$. Therefore, we can evaluate the syndrome of $\boldsymbol{t}$ as:

$$B^t\boldsymbol{t} \equiv B^t(\boldsymbol{v}+\boldsymbol{e}) \equiv B^t\boldsymbol{v} + B^t\boldsymbol{e} \equiv B^t\boldsymbol{e} \bmod 1.$$

and notice that $B^t\boldsymbol{v} \equiv \mathbf{0} \bmod 1$ since $\boldsymbol{v}$ is a point in the dual[10]. This allows us to focus on the syndrome $\boldsymbol{e}$ only and not on the particular target $\boldsymbol{t}$. In particular $\boldsymbol{e}$ belongs to the coset

$$\boldsymbol{t} + \mathcal{L}(B)^* = \{\boldsymbol{t} + \boldsymbol{x} : \boldsymbol{x} \in \mathcal{L}(B)^*\}.$$

So decoding $\boldsymbol{t}$ and recovering the shortest vector $\boldsymbol{e}$ in the coset are equivalent, using this dual lattice formulation.

10: Remark that, algebraically, $\boldsymbol{v} = B^{-t}\boldsymbol{x}$ for some $x \in \mathbb{Z}^n$

Definition 23 (*Syndrome decoding problem* (SDP), *dual lattice version*)

Given $\mathcal{L}(B)$ and some target $\boldsymbol{t} \in \mathcal{L}^*$, recover the shortest $\boldsymbol{e} \in \mathbb{R}^n$ such that $B \cdot \boldsymbol{e} \equiv B^{-t} \cdot \boldsymbol{t} \bmod 1$.

This also suggests that, in general, the dual lattice and its basis can be used to decode points in the primal (and vice versa), see, e.g., Section 4.3.2. We will discuss more in Section 4.5.1.1. Additionally, this perspective will be very useful when dealing with *short-secret LWE* and in Chapter 4, where we will a more general definition of SDP.

2.1.6 Discrete Gaussian distribution

The Discrete Gaussian distribution plays a fundamental role in lattice-based cryptography: it is the core of most of the average hardness proofs, including the ones for the well-known Learning with errors and Short integer solution – that will be presented later – and in general for all primitives that involve sampling preimages from a lattice [GPV08; MP12].The adoption of the discrete Gaussian distribution was largely motivated by attacks such as the one by Nguyen and Regev [NR06], which showed that sampling preimages uniformly from a parallelepiped leaks the trapdoor; more details follow.

[GPV08]: Gentry, Peikert, and Vaikuntanathan, *'Trapdoors for hard lattices and new cryptographic constructions'* . In: STOC '08

[MP12]: Micciancio and Peikert, *'Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller'* . In: EUROCRYPT '12

[NR06]: P. Q. Nguyen and Regev, *'Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures'* . In: EUROCRYPT '06

We will not discuss how signatures are implemented using lattice-based cryptography, but we will give a quick intuition on this attack. Think of a message $\boldsymbol{m}$ to be signed: the first procedure is to hash it as $H(\boldsymbol{m}) \in \mathbb{R}^n$ into a fixed-size message, then sample a point $\boldsymbol{t}$ that is close to $h(\boldsymbol{m})$. The public key is a bad lattice basis, but that allows us to check the distance between $\boldsymbol{t}$ and $H(\boldsymbol{m})$: if it is small, then the signature is

valid. At the core of this approach there is the impossibility of efficiently rounding $H(\boldsymbol{m})$ to the closest lattice point $\boldsymbol{t}$ without having access to a good basis of the lattice. In literature we refer to this type of approach as hash-then-sign based on CVP_γ, and some (more secure) examples are, e.g., [Fou+18; Bos+23].

[Fou+18]: Fouque, Hoffstein, Kirchner, Lyubashevsky, et al. (2018), *'Falcon: Fast-Fourier lattice-based compact signatures over NTRU'* . In: Submission to the NIST's post-quantum cryptography standardization process

[Bos+23]: J. W. Bos, Bronchain, Ducas, Fehr, et al. (2023), HAWK . In: Submission to the NIST's post-quantum cryptography standardization process

Learning a parallelepiped. Think of a signature as a BDD target in the span of some lattice $\mathcal{L}$. A valid signature $\boldsymbol{t}$ is the solution to CVP, given as input $H(\boldsymbol{m})$. This procedure is represented in Figure 37.

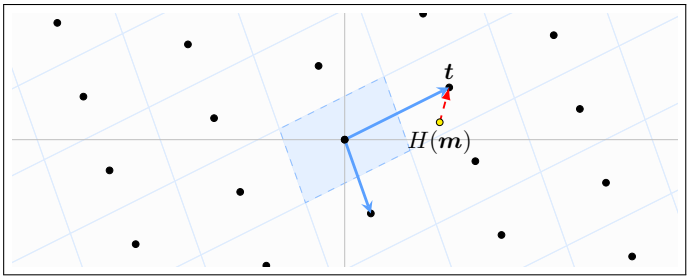


Figure 37: Some hashed message $H(\boldsymbol{m})$ and its closest lattice point $\boldsymbol{t}$

Observe that, by collecting a decent number of signatures, we can bring the distance to the origin and draw the hashed messages close to the origin (Figure 38)

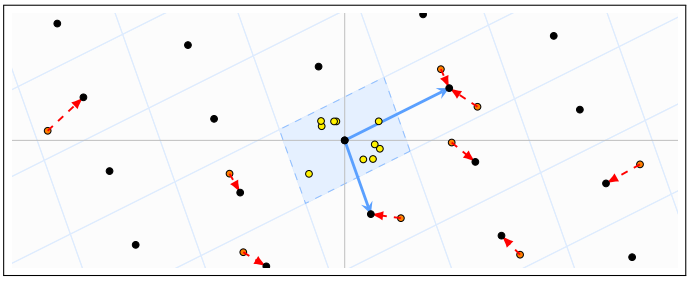


Figure 38: Reducing a number of hashed messages $H(\boldsymbol{m})$ to the origin

Nguyen and Regev [NR06] showed that, by having access to a not so large number of signatures (around 400 for NTRUSign under standard parameters), it is possible to "reconstruct" the good basis of the lattice by "learning a parallelepiped", as illustrated in Figure 39.

They were able to apply this attack to the NTRUSign [Hof+03] and GGH [GGH97] schemes.

One solution, and the one taken in (almost all) subsequent lattice-based cryptography works, is to use Gaussians. Before

[Hof+03]: Hoffstein, Howgrave-Graham, Pipher, Silverman, et al., *'NTRUSign: Digital Signatures Using the NTRU Lattice'* . In: CT-RSA '03

[GGH97]: Goldreich, Goldwasser, and Halevi, *'Public-key cryptosystems from lattice reduction problems'* . In: CRYPTO '97

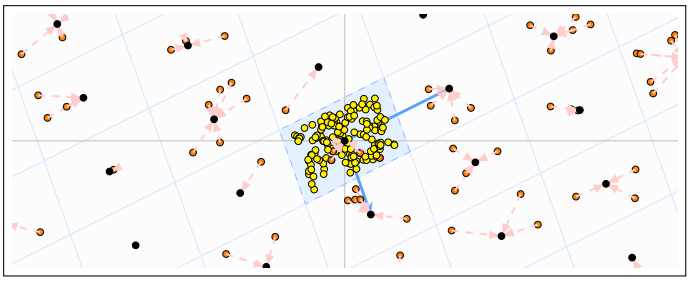


Figure 39: Reducing a number of some hashed message $H(\boldsymbol{m})$ to the origin

going into details: instead of rounding $\boldsymbol{t}$ to the closest lattice point, we use a discrete Gaussian sampler, centered in $\boldsymbol{t}$, as the new "rounding" algorithm. The intuition is given in Figure 40.

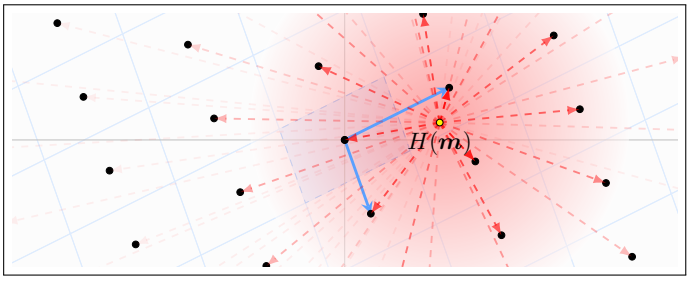


Figure 40: Randomizing the rounding of an hashed message $H(\boldsymbol{m})$ using a discrete Gaussian

As a result, the previous attack can not be applied anymore, as reducing the vectors in the origin will simply draw a sphere of radius that equals the $\sigma > 0$ used to randomize the rounding (which is going to be public information). Nevertheless, the length $H(\boldsymbol{m}) - \boldsymbol{t}$ can still be made small so that verification of signatures can be still valid.

Some formal definitions. Let $\rho_{\sigma,c}(x) := e^{-\pi(c-x)^2/\sigma^2}$ be the Gaussian function over $\mathbb{R}$ with parameter $\sigma > 0$ and center $c \in \mathbb{R}$. Notice that, if avoided, we assume $c = 0$. We define the *discrete Gaussian distribution* by restricting the continuous Gaussian distribution to a discrete lattice.

Definition 24 (*Discrete Gaussian distribution*)

For any lattice $\mathcal{L}(B)$, center $\mathbf{c} \in \mathbb{R}^n$ and Gaussian width $\sigma > 0$, the discrete Gaussian distribution *is defined as*

$$\mathcal{D}_{\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\sum_{y=-\infty}^{\infty} \rho_{\sigma,c}(y)}, \quad \forall x \in \mathbb{Z}.$$

In particular, we take the continuous Gaussian evaluated over $\mathbb{Z}$ and we "normalize" it using $\sum_{y=-\infty}^{\infty} \rho_{\sigma,c}(y)$ to make sure that the sum over all possible x is equal to 1. A representation of such distribution is given by the yellow points in Figure 41.

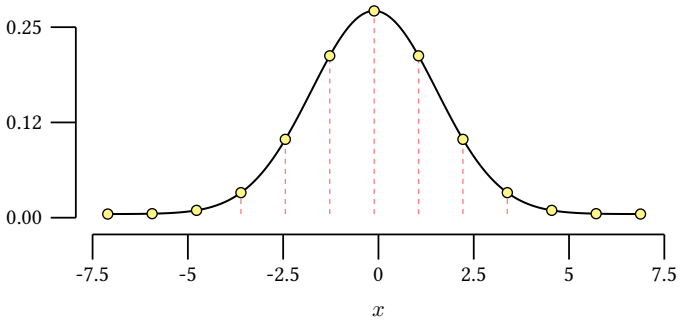


Figure 41: The continuous (black) and the discrete (yellow) Gaussian functions

One might think that obtaining this distribution is as easy as sampling from the continuous Gaussian and rounding to the closest integer, although the resulting distribution is slightly different. We avoid technical details and simply show that by doing so, the two sets of points do not perfectly match (Figure 42).

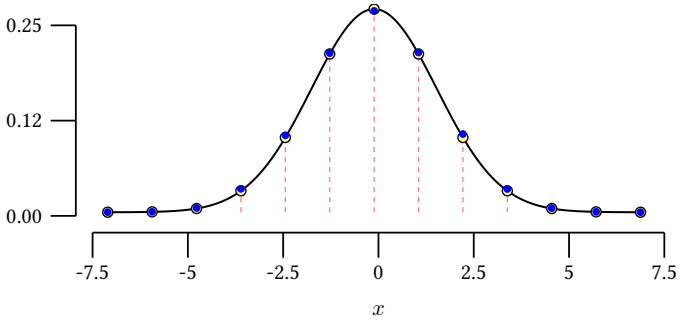


Figure 42: The continuous (black), the discrete (yellow) and the rounded (blue) Gaussian functions

Sometimes considering this simpler "rounded" distribution can be useful, as for large parameters $\sigma > 0$ it behaves very similarly to the discrete one – nonetheless, we will always assume discrete Gaussians. Indeed, we have an efficient algorithm that allows us to sample from a discrete Gaussian that can be used as building block for later reasonings.

Lemma 8 (from [Bra+13, Lemma 2.3]**)**

There is a probabilistic polynomial time algorithm that, given a basis B of a lattice $\mathcal{L}(B)$, a center $\mathbf{c} \in \mathbb{R}^m$ and a parameter $\sigma \geq \|\tilde{B}\| \cdot \sqrt{\ln(2m+4)/\pi}$, returns a sample distributed as $\mathcal{D}_{\mathcal{L}(B),\sigma,\mathbf{c}}$.

Notice that this lemma also tells us that the σ must be chosen according to $\|\tilde{B}\|$, which is a representative of the quality of the basis B. As explained in Section 2.1.4, the longest GSO vector in bad bases is larger than in good bases, meaning that in order to sample somewhat short vectors, i.e., using small parameter σ, requires to have access to a good basis. This is also compatible with the previous solution presented to fix the "learning a parallelepiped" attack, namely only someone that has the knowledge of a good basis for the lattice is able to perform a rounding (namely sample according to a discrete Gaussian centered in $H(\boldsymbol{m})$), while for someone who knows the bad basis only (public information), there is no algorithm to efficiently sample a close vector (or to forge a signature, in the case of signature schemes).

This distribution is very interesting as (i) it can be implemented coordinate-wise, namely all methods to sample the Gaussian distribution on an n-dimensional lattice use, as a building block, a procedure to sample the 1-dimensional Gaussian distribution on $\mathbb{Z}$ (i.e., it is highly parallelizable) and (ii) by working modulo the lattice, it approximates a uniform error distribution – regardless of the basis. Let us for a moment forget about the discrete distribution and consider the one dimensional lattice $\mathbb{Z} \subset \mathbb{R}$ (Figure 43).

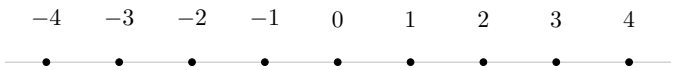


Figure 43: The one dimensional lattice $\mathbb{Z}$

Suppose to sample a value according to the $\rho(\boldsymbol{x})_\sigma$ distribution, and observe the "error" (or syndrome) with respect to the $\mathbb{Z}$ lattice – since we are working in this lattice we can simply consider the value mod 1 (Figure 44). It can be shown that

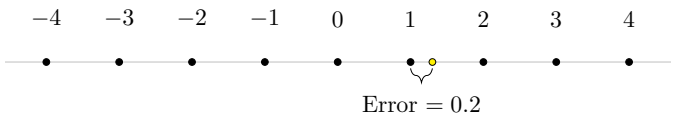


Figure 44: The one dimensional lattice $\mathbb{Z}$ and a Gaussian sample 1.2 (in yellow)

by taking sufficiently large $\sigma > 0$, the distribution of errors obtained this way is statistically close to the uniform one in $[0, 1]$. This is a pretty simple result that can easily be observed by sampling, say one thousand samples from a Gaussian with width $\sigma = 2$, reducing them modulo 1 and plotting the sorted results in $[0, 1]$. The distribution will be "visually" uniform-looking.

By extending this observation to arbitrary lattices, and by taking a more formal path, we can introduce the *smoothing parameter* $\eta_\varepsilon(\mathcal{L})$, introduced by Micciancio and Regev [MR07]. This parameter has a key role in worst-to-average case reductions for lattice-based cryptography [Chu+13]. Informally, it represents the minimum amount of Gaussian noise σ that "smooths out" the discrete structure of the lattice (up to a statistical error ε). Formally, it is the smallest $\sigma > 0$ such that the Gaussian mass[11]

$$\rho_{1/\sigma}(\boldsymbol{x}) := e^{-\pi\sigma^2\|\boldsymbol{x}\|^2}, \quad \forall \boldsymbol{x} \in \mathcal{L}^* \backslash \{\mathbf{0}\},$$

is at most $\varepsilon > 0$.This condition is equivalent to: the distribution of a continuous Gaussian of width σ, reduced modulo $\mathcal{L}$, has point-wise probability density within a $(1 \pm \varepsilon)$ factor of that of the uniform distribution over $\mathbb{R}^n/\mathcal{L}$. The usage of points belonging to the dual lattice comes from the fact that they provide a convenient way to detect if the structure is "smooth", but we refer to the original paper for more details [MR07].

[MR07]: Micciancio and Regev (2007), *'Worst-Case to Average-Case Reductions Based on Gaussian Measures'* . In: SIAM Journal on Computing

[Chu+13]: Chung, Dadush, F.-H. Liu, and Peikert, *' On the Lattice Smoothing Parameter Problem '* . In: CCC '13

11: The Gaussian mass of a set A is defined as $\rho_s(A) = \sum_{x\in A} p_s(x)$

Consider the lattice in Figure 45 and the Gaussian distributions around each lattice point.

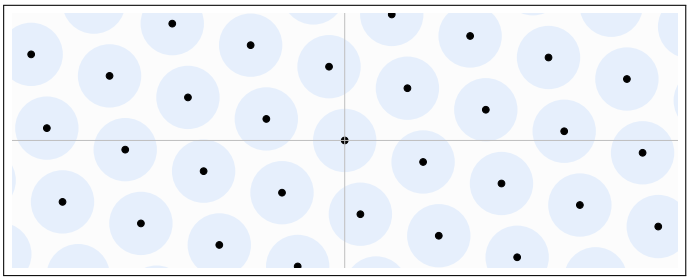


Figure 45: A lattice and some Gaussian distributions around each lattice point

By increasing the radius of the Gaussian width we can make the spheres overlap with each other (Figure 46). This means

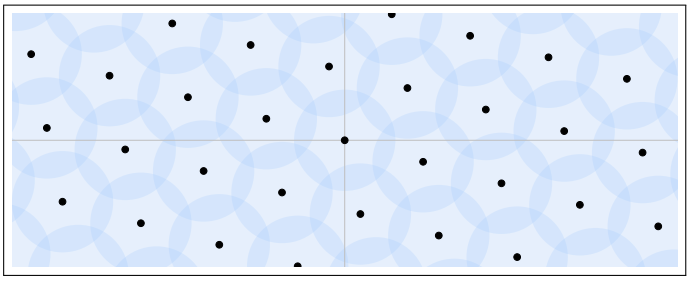


Figure 46: A lattice and some larger Gaussian distributions around each lattice point

that the probability of sampling a point in $\mathbb{R}^n$ that lies in darker areas is larger. Interestingly, if we keep increasing

the Gaussian width, we will reach some point in which the probability of every point to be sampled is (somewhat close to) uniform. It is not possible to obtain a real uniform distribution for any finite width, although we can obtain something that at some point is computationally indistinguishable (Figure 47).

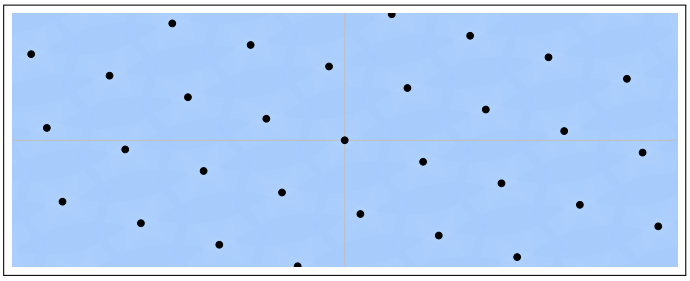


Figure 47: A lattice and some even larger Gaussian distributions around each lattice point

Definition 25 (*Smoothing parameter*)

For any lattice $\mathcal{L}(B)$, and $\varepsilon > 0$, we define the smoothing parameter *$\eta_\varepsilon(\mathcal{L}(B))$ of $\mathcal{L}(B)$ as the minimal $\sigma > 0$ such that*

$$\rho_{\mathcal{L}(B),1/\sigma}(\mathbb{Z}^n) \leq 1 + \varepsilon$$

It turns out that this parameter η_ε is usually logarithmically larger than the λ_n length of a lattice. In practice, the width of a Gaussian sample is typically not larger than $\log(n) \cdot \lambda_n\sqrt{n}$. So, in expectation, each component r_i of any sample $\boldsymbol{r}$ from this Gaussian noise is such that $r_i \approx \eta_\varepsilon \leq \log(n) \cdot \lambda_n$.

For concrete cryptographic applications, since we work in discretized spaces (think of q-ary lattices for LWE), we can not use classical Gaussian distributions. For this reason we use discrete Gaussians, and by putting together these two, we can (with much work) prove that BDD targets sampled according to this distribution – and reduced in $\mathcal{P}(\mathcal{L})$ are uniform-looking and indistinguishable from random noise – except if one has access to a good basis for the lattice so that they can decode efficiently.

We can study the norm of these samples using the following lemma.

Lemma 9 (*Tailbound*, [MR07, Lemma 4.4])

For any lattice $\mathcal{L}(B)$, vector $\boldsymbol{c} \in \mathbb{R}^n$ and reals $0 < \varepsilon < 1$, $\sigma \geq \eta_\varepsilon(\mathcal{L}(B))$ we have

$$\Pr_{\boldsymbol{x} \sim \mathcal{D}_{\mathcal{L}(B),\sigma,\boldsymbol{c}}} [\|\boldsymbol{x} - \boldsymbol{c}\| > \sigma\sqrt{n}] \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}.$$

So the syndrome $\boldsymbol{c} - \boldsymbol{x}$ has norm at most $\sigma\sqrt{n}$ except with exponentially decreasing probability. The main usage of this tool is that it allows one to sample syndromes with somewhat short norm that are, when reduced modulo the lattice, uniform-looking.

Practical sampling. Although Lemma 8 is an important theoretical tool, in most applications we do not really want to use the *real* discrete Gaussian distribution, but rather some distribution that is very close to it. For that we have multiple choices, and we will quickly review some of them.

One of the first practical examples is due to Gentry, Peikert and Vaikuntanathan [GPV08], who proposed an algorithm (see [GPV08, Theorem 4.1]) whose output is statistically close to $\mathcal{D}_{\mathcal{L},\sigma,\boldsymbol{c}}$. They define the subroutine Sample$\mathbb{Z}$, which samples from $\mathcal{D}_{\mathbb{Z}}$, and ingeniously use a randomized version of Babai's nearest plane algorithm to extend the sampler to arbitrary lattices. Their algorithm, presented in [GPV08, Section 4.2], can be seen as a randomized version of that decoding algorithm. In particular, starting from Algorithm 3, one could define a discrete Gaussian sampler by tweaking line 6 and by "randomly" round using Sample$\mathbb{Z}$. We remark that a first version of this randomized rounding firstly appeared in [Kle00], but its goal was still applied to solving CVP.

A couple of years later, Peikert [Pei10] proposed another sampler that, by splitting the sample in two phases, offline and online, is able to accelerate the sampling by means of offline precomputations. In the subsequent years, there has been a proliferation of different approaches to discrete Gaussian samplers (e.g., [MP12; Buc+14; Duc+13; DLP14a; MW17]); we refer to one of the most recent works [Gen+20] for further information about discrete Gaussians.

Usages of discrete Gaussians. In general, discrete Gaussian distributions unlock proofs for many lattice-based trapdoors, but it also enables many advanced cryptographic primitives on lattices, including, e.g., Identity-based encryption

[GPV08]: Gentry, Peikert, and Vaikuntanathan, *'Trapdoors for hard lattices and new cryptographic constructions'* . In: STOC '08

[Kle00]: Klein, *'Finding the closest lattice vector when it's unusually close'* . In: SODA '00

[Pei10]: Peikert, *'An Efficient and Parallel Gaussian Sampler for Lattices'* . In: CRYPTO '10

[MP12]: Micciancio and Peikert, *'Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller'* . In: EUROCRYPT '12

[Buc+14]: Buchmann, Cabarcas, Göpfert, Hülsing, et al., *'Discrete Ziggurat: A Time-Memory Trade-Off for Sampling from a Gaussian Distribution over the Integers'* . In: SAC '13

[Duc+13]: Ducas, Durmus, Lepoint, and Lyubashevsky, *'Lattice Signatures and Bimodal Gaussians'* . In: CRYPTO '13

[DLP14a]: Ducas, Lyubashevsky, and Prest, *'Efficient Identity-Based Encryption over NTRU Lattices'* . In: ASIACRYPT '14

[MW17]: Micciancio and Walter, *'Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time'* . In: CRYPTO '17

[Gen+20]: Genise, Micciancio, Peikert, and Walter, *'Improved Discrete Gaussian and Subgaussian Analysis for Lattice Cryptography'* . In: PKC '20

[DLP14b], Program obfuscation [CLW25], Zero-knowledge proofs [LNP22] and Polynomial commitments [Cin+24].

[DLP14b]: Ducas, Lyubashevsky, and Prest, *'Efficient identity-based encryption over NTRU lattices'* . In: ASIACRYPT '14

[CLW25]: Cini, Lai, and Woo, *'Lattice-Based Obfuscation from NTRU and Equivocal LWE'* . In: CRYPTO '25

[LNP22]: Lyubashevsky, N. K. Nguyen, and Plançon, *'Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General'* . In: CRYPTO '22

[Cin+24]: Cini, Malavolta, N. K. Nguyen, and Wee, *'Polynomial Commitments from Lattices: Post-Quantum Security, Fast Verification and Transparent Setup'* . In: CRYPTO '24

2.1.7 q-ary lattices

In cryptography, we are very much interested in a very specific family of lattices, that goes under the name of *q-ary lattices*.

Definition 26 (*q-ary lattice*)

A lattice $\mathcal{L}$ is said to be q-ary *if*

$$q\mathbb{Z}^m \subseteq \mathcal{L} \subseteq \mathbb{Z}^m. \tag{2.14}$$

Intuitively, any lattice from this family has the property of being tiled modulo q throughout the $\mathbb{Z}^m$ space. By considering integer q-ary lattices, we can easily perform computations with them using standard integer arithmetic in $\mathbb{Z}_q$, this also makes it easier to represent and study problems over such lattices, indeed most of the cryptographic primitives based on lattices are built upon this family of lattices. One way to build such a lattice is to take some matrix $A \in \mathbb{Z}^{m\times n}$ modulo q and consider the "image" of the A matrix.

Definition 27 (*Row-generated lattice $\mathcal{L}_q$*)

Fix positive integers $n \leq m \leq q$ and $A \in \mathbb{Z}_q^{m\times n}$. Define the row-generated lattice $\mathcal{L}_q(A)$ *as:*

$$\mathcal{L}_q(A) := \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} \equiv A\boldsymbol{x} \bmod q \text{ for some } \boldsymbol{x} \in \mathbb{Z}_q^n\}.$$

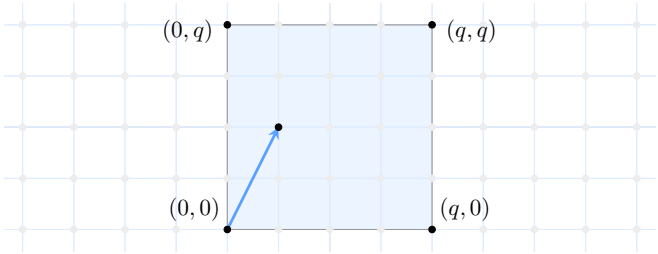
In other words, the lattice is composed of all possible $\boldsymbol{y} \in \mathbb{Z}^m$ such that there exists some $\boldsymbol{x}$ that generates it through A. We show an example on how to build such a lattice in Example 2.

Example 2 (*Building a q-ary lattice $\mathcal{L}_q(A)$*)

Let us fix some $A \in \mathbb{Z}_q^{m\times n}$, given $m := 2, n := 1$ and $q := 4$. So, for instance

$$A := \begin{pmatrix} 1 \\ 2 \end{pmatrix} \in \mathbb{Z}_4^{2\times 1}$$

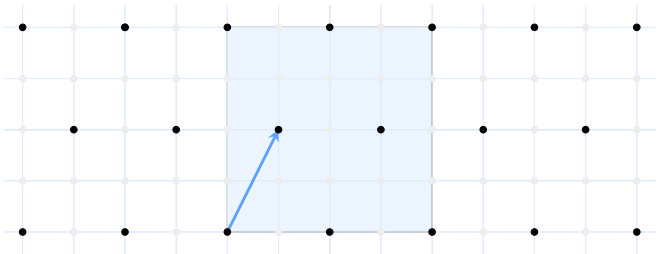
Now, we can plot the vector(s) of A such inside the fundamental region of $q\mathbb{Z}^m$.



At this point, simply compute all the integer combinations modulo $q = 4$. For instance:

- $A \equiv (2, 1)^t$
- $A \cdot 2 \equiv (2, 0)^t$
- $A \cdot 3 \equiv (3, 2)^t$

Then, by tiling all these points modulo q, we obtain the lattice, which can also be seen as $\mathcal{L}_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$:



Observe that $\mathcal{L}_q(A)$ naturally lives inside $\mathbb{Z}^m$.

Note that A itself is not a lattice basis, we will call it a *q-ary basis*. On the other hand, we can also define another q-ary lattice, picking now a different set of points.

Definition 28 (*Parity check lattice $\mathcal{L}_q^\perp$*)

Fix positive integers $n \leq m \leq q$ and $A \in \mathbb{Z}_q^{m \times n}$. Define the parity check lattice *$\mathcal{L}_q^\perp(A)$ as:*

$$\mathcal{L}_q^\perp(A) = \{\boldsymbol{x} \in \mathbb{Z}^m : A^t\boldsymbol{x} \equiv \mathbf{0} \bmod q\} \subseteq \mathbb{Z}^m. \tag{2.15}$$

We use the *parity check lattice* term, borrowed from coding theory, but we will also call this the SIS lattice later. In particular, $\mathcal{L}_q^\perp(A)$ contains all the solutions to the system of linear equation $A^t\boldsymbol{x} \equiv \mathbf{0}$.

A digression on systematic forms. Without loss of generality, we will mostly consider matrices of the form (called *systematic form* or *Hermite normal form*):

$$\begin{pmatrix} A \\ I_n \end{pmatrix} \in \mathbb{Z}_q^{m \times n}, \tag{2.16}$$

for some $A \in \mathbb{Z}_q^{(m-n) \times n}$. We can indeed assume that any uniformly sampled $B \sim \mathbb{Z}_q^{m \times n}$, using sufficiently large parameters, will contain n linearly independent row vectors over $\mathbb{Z}_q$, so we can move them to the bottom part of the matrix and write

$$B = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \in \mathbb{Z}_q^{m \times n},$$

for some invertible $B_2 \in \mathbb{Z}_q^{n \times n}$. So we can write $B \cdot B_2^{-1}$ to obtain:

$$B \cdot B_2^{-1} = \begin{pmatrix} B_1 \cdot B_2^{-1} \\ I_n \end{pmatrix} \in \mathbb{Z}_q^{m \times n},$$

which is now in the shape of Equation 2.16, with $A = B_1 \cdot B_2^{-1}$. Notice that all procedures involved (i.e., permuting vectors, multiplication by B_2^{-1}) do not change the underlying lattice, as highlighted by the following lemmas.

Lemma 10 (*q-ary lattices under row permutations*)

Let A' be a row permutation of A (i.e., $A' = PA$ for some permutation matrix P), then $\mathcal{L}_q^\perp(A') = P \cdot \mathcal{L}_q^\perp(A)$.

In particular, they differ only by a coordinate permutation, and their properties (e.g. determinant, successive minima) are the same.

Lemma 11 (*q-ary lattices under invertible transformations*)

Let $B \in \mathbb{Z}_q^{n \times n}$ be invertible. Then $\mathcal{L}_q^\perp(A \cdot B) = \mathcal{L}_q^\perp(A)$.

Proof. Recall that $\mathcal{L}_q^\perp(A) = \{\boldsymbol{x} \in \mathbb{Z}^m : A^t \boldsymbol{x} \equiv \mathbf{0} \bmod q\}$. Then $\boldsymbol{x} \in \mathcal{L}_q^\perp(A \cdot B)$ if and only if $(A \cdot B)^t \boldsymbol{x} \equiv \mathbf{0}$. We can reformulate ad as $B^t A^t \boldsymbol{x} \equiv 0 \bmod q$. Since B is invertible modulo q, so is B^t, and thus:

$$B^t A^t \boldsymbol{x} \equiv \mathbf{0} \iff A^t \boldsymbol{x} \equiv \mathbf{0} \bmod q.$$

Therefore, $\mathcal{L}_q^\perp(A \cdot B) = \mathcal{L}_q^\perp(A)$. □

Remark 1 (☞)

For the sake of clarity, we will always consider q-ary basis of size $m \times n$, with $m > n$, and simply use the transpose A^t when referring

to $\mathcal{L}_q^\perp(A)$. In literature, sometimes it is possible to find usages of A, without transposition, but with swapped dimension $n \times m$. In our case, we fix the dimension to always be $m \times n$, and transpose if required.

Notice that $\mathcal{L}_q^\perp(A)$ is a lattice since the set of solutions of an homogeneous system of equations is closed under addition[12], so every point is a possible solution – as it will be shown later, recovering the "shortest" solution is considered hard as it implies to solve SVP on such lattice. There is a strong relation between these two lattices, and this observation is the key intuition to better understand how modern LWE-based schemes work.

12: In general, given $A, \boldsymbol{x}_1$ and $\boldsymbol{x}_2$ such that $A\boldsymbol{x}_1 = 0$ and $A\boldsymbol{x}_2 = 0$, then $A(\boldsymbol{x}_1 + \boldsymbol{x}_2) = A\boldsymbol{x}_1 + A\boldsymbol{x}_2 = 0$

Lemma 12 (*Duality between q-ary lattices*)

Fix positive integers $n \leq m \leq q$ and $A \in \mathbb{Z}_q^{m \times n}$. Then

$$\mathcal{L}_q(A)^* = \frac{1}{q}\mathcal{L}_q^\perp(A),$$

or, equivalently

$$\mathcal{L}_q^\perp(A) = q \cdot \mathcal{L}_q(A)^*.$$

Proof. We know that any point $\boldsymbol{x} \in \mathcal{L}_q^\perp(A)$ is such that the coordinates of $A^t\boldsymbol{x}$ are all multiples of q. If we perform an inner product between any point $\boldsymbol{x}$ of $\mathcal{L}_q^\perp$ and $\boldsymbol{y} \in \mathcal{L}_q$, we obtain:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle \equiv \langle \boldsymbol{x}, A\boldsymbol{x}' \rangle \equiv \langle A^t\boldsymbol{x}, \boldsymbol{x}' \rangle \equiv \langle \boldsymbol{0}, \boldsymbol{x}' \rangle \equiv 0 \bmod q.$$

So they are dual, up to some scaling factor q. □

For convenience, we will call the scaled version *q-dual*, e.g., the q-dual of $\mathcal{L}_q(A)$ is $\mathcal{L}_q^\perp(A)$, and vice versa.

Building an actual (canonical) lattice basis. We saw that such lattices are usually generated by $A \in \mathbb{Z}_q^{m \times n}$, but not as lattice bases. We now want to answer the question about how to construct a full-rank lattice basis for the $\mathcal{L}_q(A)$ and $\mathcal{L}_q^\perp(A)$ lattices.

Given a systematic form matrix $A \in \mathbb{Z}_q^{m \times n}$ containing $\bar{A} \in \mathbb{Z}_q^{(m-n) \times n}$ as a submatrix in the first $m - n$ rows, we can easily build an m-dimensional full-rank lattice basis B for $\mathcal{L}_q^\perp(A)$ as

$\mathcal{L}_q^\perp(A) = \mathcal{L}(B)$, with

$$B := \begin{pmatrix} 0 & I_{m-n} \\ qI_n & -\bar{A}^t \end{pmatrix} \in \mathbb{Z}_q^{m \times m}, \tag{2.17}$$

since every lattice point $\boldsymbol{v} \in \mathcal{L}(B)$ can be written as:

$$\boldsymbol{v} \equiv B\boldsymbol{x} \equiv \begin{pmatrix} 0 & I_{m-n} \\ qI_n & -\bar{A}^t \end{pmatrix} \boldsymbol{x} \equiv \begin{pmatrix} \boldsymbol{x}_2 \\ -\bar{A}^t\boldsymbol{x}_2 + q\boldsymbol{x}_1 \end{pmatrix} \in \mathbb{Z}_q^m,$$

for some $\boldsymbol{x} := \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} \in \mathbb{Z}_q^m$, with $\boldsymbol{x}_1 \in \mathbb{Z}_q^n$ and $\boldsymbol{x}_2 \in \mathbb{Z}_q^{m-n}$. To double check, we can see whether $\boldsymbol{v}$ respects Definition 28 by verifying if $A^t\boldsymbol{v} = \mathbf{0} \bmod q$ as follows:

$$A^t\boldsymbol{v} \equiv [\bar{A}^t \mid I_n] \cdot \boldsymbol{v} \equiv \bar{A}^t\boldsymbol{x}_2 - \bar{A}^t\boldsymbol{x}_2 + q\boldsymbol{x}_1 \equiv q\boldsymbol{x}_1 = \mathbf{0} \bmod q.$$

Since we know that $\mathcal{L}_q^\perp(A)$ and $\mathcal{L}_q(A)$ are dual, up to scaling by q, we can also easily find a lattice basis for $\mathcal{L}_q(A)$ simply as B^{-t}, leading to $\mathcal{L}_q(A) = \mathcal{L}(qB^{-t})$, where:

$$qB^{-t} = \begin{pmatrix} qI_{m-n} & \bar{A} \\ 0 & I_n \end{pmatrix} \in \mathbb{Z}_q^{m \times m}. \tag{2.18}$$

2.2 Short integer solution

In this and the next section we will introduce two average-case lattice problems defined over inputs sampled from some probability distribution. This is different from previously presented problems (e.g., SVP, CVP, and so on), which are standard complexity theory problems that are tackled from a worst case point of view.

The Short integer solution (SIS) [Ajt96] is one of the most important problems in lattice-based cryptography.

[Ajt96]: Ajtai, *'Generating hard instances of lattice problems (extended abstract)'* . In: STOC '96

Definition 29 (*Short integer solution*)

For a security parameter λ, positive integer modulus $q = \lambda(q)$, dimensions $n = \lambda(n), m = \lambda(m)$, and a norm bound $\beta > 0$, the $\mathrm{SIS}_{n,m,q,\beta}$ *problem is defined as: given a uniformly random $A \in \mathbb{Z}^{m \times n}$, find a nontrivial "short" solution $\boldsymbol{z} \in \mathbb{Z}^m$ such that $A^t\boldsymbol{z} \equiv \mathbf{0} \in \mathbb{Z}^n$ and $z_i \leq \beta$ for all $z_i \in \boldsymbol{z}$.*

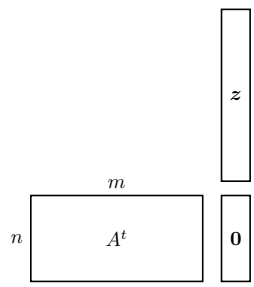


Figure 48: Shape of the matrices used in SIS

Before we anticipated that $\mathcal{L}_q^\perp(A)$ is sometimes referred to as the SIS lattice, and this is precisely why. Indeed, $\mathcal{L}_q^\perp(A)$ contains all the solutions $A^t\boldsymbol{z} \equiv \mathbf{0} \bmod q$, and the SIS problem intuitively asks to recover a bounded short vector in $\mathcal{L}_q(A)^\perp$.

We introduce two definitions that will simplify later arguments.

Definition 30 (*β-bounded vector*)

A vector $\boldsymbol{v}$ is β-bounded, for $\beta > 0$, if each $v_i \leq \beta$.

In simple words, a β-bounded vector does not contain any element larger than β.

Definition 31 (*β-bounded distribution*)

A distribution ensemble $\{\chi_n\}$ supported over the integers is called β-bounded if

$$\Pr_{x \sim \chi}[|x| > \beta] = \text{negl}(n) \tag{2.19}$$

We will use the notation $\boldsymbol{x} \sim [\beta]$ to refer to a vector whose entries are sampled from a uniform β-bounded distribution. Notice that finding *one* solution to $A^t\boldsymbol{z} = \mathbf{0}$ is easy, by simply using Gaussian elimination. On the other hand, finding some β-bounded solution, seems like a very difficult task. Given $f_A(\boldsymbol{z}) = A^t\boldsymbol{z} \bmod q$, with input some β-bounded $\boldsymbol{z}$:

- if $\beta < \lambda_1/2$ then the solution is unique and f is injective[13],
- if $\beta > \lambda_1/2$ then the solution is not unique and f is not injective,
- if $\beta \geq \mu$, with μ being the covering radius of the lattice (Definition 10) then f is surjective (see Figure 21),
- if $\beta \gg \mu$ then f is almost uniform over $\mathbb{Z}_q^n$ (see a spoiler in Figure 47)[14].

13: This corresponds, as it will be shown later, to the Learning with errors setup

14: This corresponds to the SIS setting

In general, this can be seen as "sampling" a random short target from the origin, and reduce it modulo the lattice. In the case of q-ary lattices, this corresponds to the $\bmod q$ operation.

Remark 2 (☞)

Reducing (hiding) a short vector modulo the lattice is the unique one-way function used in lattice-based cryptography [Pei13].

How many short vectors. As SIS is equivalent to looking for short vectors in $\mathcal{L}_q^\perp(A)$, it can be useful to roughly quantify how many of them we can find in a sphere of radius β. Additionally, these prove that these short vectors actually exist. We now present a couple of interesting lemmas (which are not exhaustive), which can guide the choice of β with respect to the modulus q.

Lemma 13 ([Lyu24, Lemma 2])

For any prime q and any $\boldsymbol{t} \in \mathbb{Z}_q^n \setminus \{0\}$,

$$\Pr_{\mathbf{A} \sim \mathbb{Z}_q^{m \times n}} \left[\exists \boldsymbol{z} \in [\beta]^{n+m} \text{ such that } [A^t \mid I_n]\boldsymbol{z} \equiv \boldsymbol{t} \bmod q \right] \leq \frac{(2\beta+1)^{n+m}}{q^n}.$$

What this lemma states is that the number of lattice points around some target $\boldsymbol{t} \in \mathbb{Z}_q^n$ is not very large, and in order to make it larger it we either have to increase the dimension m or the radius β. This result allows us to strongly define cases when these β-bounded vectors exist and when there are not.

Lemma 14 ([Lyu24, Lemma 4])

For any q and any $A \in \mathbb{Z}_q^{m \times n}$, there exists $\boldsymbol{z}$ such that:

$$\boldsymbol{z} \in [q^{n/(n+m)}]^{n+m} \setminus \{\mathbf{0}\} \text{such that } [A^t \mid I_n]\boldsymbol{z} \equiv \mathbf{0} \bmod q.$$

By combining these lemmas, we obtain that

- By setting $\beta < \frac{1}{4} q^{n/(m+n)}$, the probability of there being a β-bounded vector in $\mathcal{L}_q^{\perp}(A)$ is less than $2^{-(m+n)}$.
- By setting $\beta = q^{n/(n+m)}$, a β-bounded vector always exists.

We refer the interested reader to Section 3 of [Lyu24], also for the proofs of the lemmas.

[Lyu24]: Lyubashevsky (2024), Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA) (Cryptology ePrint Archive, Paper 2024/1287)

2.2.1 Variants of SIS

We can define also an inhomogeneous-SIS (ISIS) instance, where the considered equation is $A^t \boldsymbol{z} \equiv \boldsymbol{y}$, for a uniformly random $\boldsymbol{y} \in \mathbb{Z}_q^n$. In particular, it is possible to show that SIS and inhomogeneous-SIS are (somewhat) equivalent. Firstly, is easy to show that SIS $\leq$ ISIS.

Lemma 15

An inhomogeneous-SIS-solver can be used to solve SIS.

Proof. Consider the last column $\boldsymbol{a}_m \in \mathbb{Z}_q^n$ of some SIS instance $A \in \mathbb{Z}_q^{n \times m}$, and observe that it is uniformly distributed. Set it as $\boldsymbol{y} := \boldsymbol{a}_m$, and call A' the rest of the A matrix. Any solution $\boldsymbol{z}'$ to $A'\boldsymbol{z}' \equiv \boldsymbol{y}$ can be used to build $\boldsymbol{z} = (\boldsymbol{z}', -1)$, that is a

solution to the first SIS instance $A\boldsymbol{z} \equiv \mathbf{0}$. □

A similar reduction, although slightly trickier, can be performed the other way round. We can thus look at both problems as having the same hardness for simplicity.

Notice that this problem defines a one-way function $f_A(\boldsymbol{z}) = A^t\boldsymbol{z} \bmod q$ [Ajt96], which can then be used to build collision-resistant hash functions [PR06], commitments [KTX08], functional commitments [CP23], signatures [GPV08], and so on. Inverting such a function implies a method to solve SIS, or to find a short vector in $\mathcal{L}_q^\perp(A)$, which we already saw being a very hard problem. Ajtai's results also connected worst to average cases, meaning that a solution to SIS can be used to solve approximate GapSVP in the worst case, with the approximation factor depending on n and β.

As an example, let us build a simple collision resistant hash function directly from SIS. Define

$$f_A(\boldsymbol{x}) : \{0,1\}^m \to \mathbb{Z}_q^n = A^t\boldsymbol{x} \bmod q$$

for any $m > n\log(q)$. Notice that the set of preimages is larger than the set of images, but recovering a preimage $\boldsymbol{x}' \in \{0,1\}^m$ that gives the same result as some other $\boldsymbol{x} \in \{0,1\}^m$ (i.e., a collision) implies a solution to SIS, since if

$$A^t\boldsymbol{z} = A^t\boldsymbol{z}' \bmod q$$

then we can fix $\boldsymbol{w} \equiv \boldsymbol{z} - \boldsymbol{z}' \in \{-1,0,1\}^m$ that is a short solution $A^t\boldsymbol{w} \equiv \mathbf{0}$. Inverting $f_A(\boldsymbol{z})$, given some output $\boldsymbol{y}$, is equal to solving SIS under the inhomogeneous regime, as the solution is not a 0-vector. Notice that, in such case, instead of looking the problem as a standard CVP problem in $\mathcal{L}_q^\perp(A)$, we can look at it as recovering a short vector in the coset $\boldsymbol{z} + \mathcal{L}_q^\perp(A)$. We formalize the different regimes on how to instantiate an SIS instance:

- Homogeneous regime: share A, recover $\boldsymbol{z}$ such that $A^t\boldsymbol{z} \equiv \mathbf{0}$.
- Inhomogeneous regime: share $(A, \boldsymbol{y})$ for some random $\boldsymbol{y} \sim \mathbb{Z}_q^m$, recover $\boldsymbol{z}$ such that $A^t\boldsymbol{z} \equiv \boldsymbol{y}$.
- Planted (or LWE) regime: share $(A, \boldsymbol{y})$, recover $\boldsymbol{z}$ such that $A^t\boldsymbol{z} \equiv \boldsymbol{y}$ for some fixed β-bounded $\boldsymbol{z} \in \mathbb{Z}_q^m$ which is unique.

Notice that, in the first two regimes, we can prove that a solution exists using probabilistic arguments, in particular using the Gaussian heuristic (refer to Heuristic 1 of [GNR10] to

[GNR10]: Gama, P. Q. Nguyen, and Regev (2010), *'Lattice Enumeration Using Extreme Pruning'* . In: Advances in Cryptology – EUROCRYPT 2010

see that the number of solutions is indeed very large). Similar reasonings can be used, in the planted regime, to prove that the solution is presumably unique as it is very unlikely that a solution, other than $\boldsymbol{z}$, exists "naturally". Another way of calling this regime is injective ISIS, or low-density SIS [Lyu12]. On the other hand, the inhomogeneous regime, as we defined it, can be referred to as surjective ISIS, or high-density ISIS. Notice that, if not specified, we will refer to surjective ISIS.

[Lyu12]: Lyubashevsky, *'Lattice Signatures without Trapdoors'* . In: EUROCRYPT '12

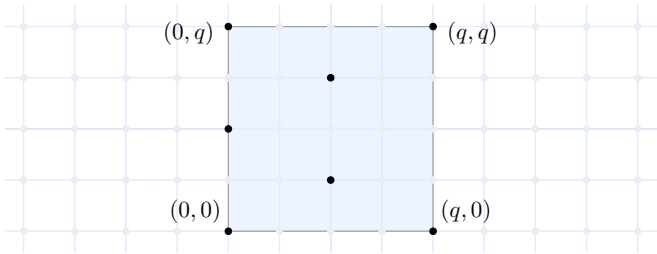
To see why SIS is a lattice problem, observe that A^t is the q-ary basis for the $\mathcal{L}_q^\perp(A)$ lattice, and $A^t\boldsymbol{z}$, with $\boldsymbol{z}$ short, should suggest a connection to SVP. Indeed, we now present an example that will also informally show how we can reduce to the hardness of solving $\mathrm{GapSVP}_{\beta\sqrt{n}}$.

Example 3 (SIS *is a lattice problem in* $\mathcal{L}_q^\perp(A)$)

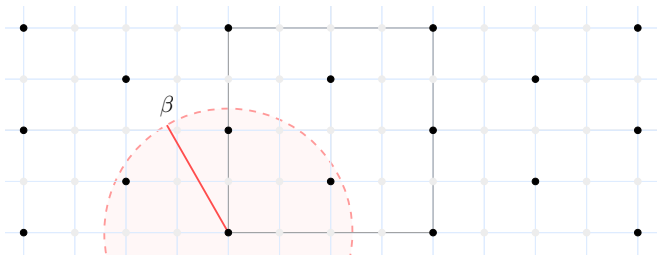
Let us fix some $A \in \mathbb{Z}_q^{m\times n}$, given $m = 2, n = 1$ and $q = 4$. So, for instance

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \in \mathbb{Z}_4^{2\times 1}$$

Now, we can plot the lattice $\mathcal{L}_q^\perp(A)$ by simply considering all the possible $\boldsymbol{z} \in \mathbb{Z}_q^n$ that satisfy $A^t\boldsymbol{z} \equiv 0 \bmod q$. It is easy to see that such solutions are $\{(0,0),(2,1),(0,2),(2,3)\}$.



The SIS problem asks to recover, among such solutions, a "short" one (with respect to $\beta \ll q$). Visually, this means that we want to find some lattice point inside the red sphere:



Notice that, contrary to the pictures, usually it holds that $\beta \ll q$ to make the problem hard, but we wanted to highlight that inside the red sphere there will be many solutions.

Nevertheless, recovering only one of them seems a very hard task.

Due to Ajtai [Ajt96], we know that solving SIS seems hard on average.

Lemma 16 (*Hardness of* SIS [Ajt96])

Solving $\mathrm{SIS}_{n,q,\beta,m}$*, that is finding some* $\boldsymbol{s} \in \mathcal{L}_q^{\perp}(A)$ *for uniformly random* $A \in \mathbb{Z}_q^{n\times m}$ *such that* $\|\boldsymbol{z}\| \leq \beta \ll q$*, is as hard as solving* $\mathrm{GapSVP}_{\beta\sqrt{n}}$ *on any* n*-dimensional lattice.*

It is possible to look at the other side of the coin, by considering the row-generated q-ary lattice $\mathcal{L}_q$.

2.3 Learning with errors

The Learning with errors (LWE) problem [Reg05] was introduced almost ten years after the breakthrough results by Ajtai. Let us first define it: fix two values $m > n$, a modulus $q = \mathrm{poly}(n)$, a small Gaussian-like error distribution χ (typically a discrete Gaussian). Now define a set of *noisy* linear equations:

[Reg05]: Regev, *'On lattices, learning with errors, random linear codes, and cryptography'* . In: STOC '05

$$\begin{aligned} \boldsymbol{a}_1 &:= \mathbb{Z}_q^n \ , \ b_1 \equiv \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle + s_1 \in \mathbb{Z}_q \\ \boldsymbol{a}_2 &:= \mathbb{Z}_q^n \ , \ b_2 \equiv \langle \boldsymbol{a}_2, \boldsymbol{x} \rangle + s_2 \in \mathbb{Z}_q \\ &\vdots \\ \boldsymbol{a}_m &:= \mathbb{Z}_q^n \ , \ b_m \equiv \langle \boldsymbol{a}_m, \boldsymbol{x} \rangle + s_m \in \mathbb{Z}_q \end{aligned} \tag{2.20}$$

We can define two "flavors" of LWE as follows:

- (Search-LWE) Given the set of all (a_i, b_i) recover either $\boldsymbol{x} \in \mathbb{Z}_q^n$ or $\boldsymbol{s} \in \chi^m$.
- (Decision-LWE) Distinguish between a uniformly random pair $(\boldsymbol{x}, y) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and a LWE sample $(\boldsymbol{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Usually, we define LWE in a more compact way simply as the pair $(A, \boldsymbol{b})$, with $\boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s} \bmod q$, $A \sim \mathbb{Z}_q^{m\times n}$, $\boldsymbol{x} \sim \mathbb{Z}_q^n$ and $\boldsymbol{s} \sim \chi^m$. Usually in literature the notation for LWE is $A\boldsymbol{s} + \boldsymbol{e}$. In our case, we use a different notation that will be more useful later (especially in Chapter 4). We now provide a formal definition for decision-LWE.

Definition 32 (*Learning with errors* ($\mathrm{LWE}_{\boldsymbol{n},\boldsymbol{m},\boldsymbol{q},\boldsymbol{\chi}}$))

For a security parameter λ*, let* $n = n(\lambda)$ *be an integer dimension,*

let $q = q(\lambda) \geq 2$ be an integer, let $m = \mathcal{O}(n \log(q))$ be the number of samples, and let $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}$. The $\text{LWE}_{n,m,q,\chi}$ problem is to distinguish with non-negligible probability (in λ) the following two distributions both returning $(A, \boldsymbol{b})$: in the first one draws $A, \boldsymbol{b}$ uniformly from $\mathbb{Z}_q^{m \times n}$ and $\mathbb{Z}_q^m$ respectively. In the second distribution one first draws $\boldsymbol{x} \in \mathbb{Z}_q^n$ uniformly, then samples A uniformly from $\mathbb{Z}_q^{m \times n}$ and $\boldsymbol{s} \in \mathbb{Z}_q^m$ with $s_i \sim \chi$, and sets $\boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s} = (\langle a_i, \boldsymbol{s} \rangle + s_i \bmod q)_i$. The $\text{LWE}_{n,q,m,\chi}$ assumption is that the $\text{LWE}_{n,m,q,\chi}$ problem is infeasible.

In order to introduce the hardness of this problem, Regev showed [Reg05] that there is a (quantum) reduction from worst case $\text{GapSVP}_{nq/\sigma}$ to search-LWE, where $\sigma > 0$ is the standard deviation of χ. For worse parameters (exponentially large in n) we also have a classical reduction [Pei09; Bra+13].

[Pei09]: Peikert, *'Public-key cryptosystems from the worst-case shortest vector problem: extended abstract'* . In: STOC '09

[Bra+13]: Brakerski, Langlois, Peikert, Regev, et al., *'Classical hardness of learning with errors'* . In: STOC '13

Theorem 4 (*Quantum hardness of LWE* [Reg05])

For $\chi = \mathcal{D}_{\mathbb{Z},\sigma}$ with $\sigma \geq \sqrt{n}/q$, and any $m = \text{poly}(n)$, the search-$\text{LWE}_{n,m,q,\chi}$ problem is hard on average (even for quantum algorithms) if $\text{GapSVP}_{nq/\sigma}$ on n-dimensional lattices is hard in the worst case for quantum algorithms.

Observe that larger errors (i.e., larger values of σ), imply smaller approximations factors, and in turn harder problems. Furthermore, there is a reduction between the two versions of LWE, namely search-LWE $\leq$ decision-LWE [MM11]. The abstract idea is that it is possible to learn one entry of $\boldsymbol{x}$ at a time with q calls to a decision-LWE oracle. Therefore, with $q \cdot n$ decision-LWE calls we can solve search-LWE as well[15]. Moreover, the reduction can also be made more efficient (only two calls) using a quantum reduction [Sud+23]. The reduction in the other direction is trivial.

[MM11]: Micciancio and Mol (2011), *'Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions'* . In: Advances in Cryptology – CRYPTO 2011

15: A more efficient reduction can be done, leading to only $O(n \log(q))$ calls to the oracle

[Sud+23]: Sudo, Tezuka, Hara, and Yoshida, *'Quantum Search-to-Decision Reduction for the LWE Problem'* . In: AFRICACRYPT '23

One might argue that, however, practical applications based on LWE typically use different distributions from the (real) discrete Gaussian $\mathcal{D}$. This happens because sampling from, e.g., a uniform distribution is much more quicker and easier for practical implementations – moreover, LWE still appears to be hard for such error distributions. However, proving the hardness of LWE using other error distributions with proper reductions seems harder.

For completeness, we also give an informal theorem regarding the (classical) hardness of LWE, although we refer to the original paper for the extensive proof [Bra+13].

[Bra+13]: Brakerski, Langlois, Peikert, Regev, et al., *'Classical hardness of learning with errors'* . In: STOC '13

Theorem 5 (*Informal,* [Bra+13, Theorem 1.1])

Solving n-dimensional LWE *with* $\text{poly}(n)$ *modulus implies an equally efficient solution to a worst-case lattice problem in dimension* $\sqrt{n}$.

Short secret LWE. An important result, which will later improve the "interpretability" of LWE, is that it can be instantiated using the same small distribution χ for both the error $\boldsymbol{s}$ and the secret $\boldsymbol{x}$. The core idea, firstly proposed in [App+09], is to take advantage of Hermite normal form [Mic01b]. Remark that we can, without loss of generality, assume that A is in the form of 2.16. In the case of LWE, we can use this transformation to introduce short-secret-LWE as follows:

[App+09]: Applebaum, Cash, Peikert, and Sahai, *'Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems'* . In: CRYPTO '09

[Mic01b]: Micciancio (2001), *'Improving Lattice Based Cryptosystems Using the Hermite Normal Form'* . In: Cryptography and Lattices

1. Consider the LWE instance $A \in \mathbb{Z}_q^{m \times n}$ and $\boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s} \in \mathbb{Z}_q^m$.
2. Write A as
$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$$
where $A_2 \in \mathbb{Z}_q^{n \times n}$ is invertible.
3. Multiply by A_2^{-1} as:
$$A \cdot A_2^{-1} = \begin{pmatrix} A_1 A_2^{-1} \\ I_n \end{pmatrix}$$
and define for convenience $A' = A_1 A_2^{-1} \in \mathbb{Z}_q^{(m-n) \times n}$.
4. This leads to a new equation for LWE, namely
$$\begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} A' \\ I_n \end{pmatrix} \cdot \boldsymbol{x}' + \begin{pmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$
where $\boldsymbol{x}' = A_2\boldsymbol{x}$ to "counter" the effect of $A \cdot A_2^{-1}$ and both $\boldsymbol{b}$ and $\boldsymbol{s}$ are split in two parts of size $(m-n)$ and n, respectively.
5. Now fix
$$L = \begin{pmatrix} I_{m-n} & -A' \\ 0 & I_n \end{pmatrix}$$
which is computable since it only uses public material, and multiply everything by L:
$$L \cdot \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} = L \cdot \begin{pmatrix} A' \\ I_n \end{pmatrix} \cdot \boldsymbol{x}' + L \cdot \begin{pmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$

6. This leads to

$$\begin{pmatrix} \boldsymbol{b}_1 - A'\boldsymbol{b}_2 \\ \boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{x}' \end{pmatrix} + \begin{pmatrix} \boldsymbol{s}_1 - A'\boldsymbol{s}_2 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$

And notice that now we have a new short secret LWE sample in the top part, as

$$\underbrace{\boldsymbol{b}_1 - A'\boldsymbol{b}_2}_{\text{new } \boldsymbol{b}} = \underbrace{A'}_{\text{new } A} \cdot \underbrace{-\boldsymbol{s}_2}_{\text{new } \boldsymbol{x}} + \underbrace{\boldsymbol{s}_1}_{\text{new } \boldsymbol{s}}$$

so the new secret key becomes $-\boldsymbol{s}_2$, and if we recover it we can also find a solution for the original instance with $\boldsymbol{x} = A_2^{-1} \cdot \boldsymbol{x}' = A_2^{-1} \cdot (\boldsymbol{s}_2 - \boldsymbol{b}_2)$.

Notice that this point of view allows us to focus on the short $\boldsymbol{s}$ vector, and not on the particular secret $\boldsymbol{x}$, similarly to moving from a target to the syndrome. As we will see later, this interpretation fully focuses on the distance between some BDD target and a lattice point in some q-ary lattice, and not on the particular target itself, therefore using the SDP assumption. As a result, we can sample the secret $\boldsymbol{x}$ from the same "short" distribution χ as $\boldsymbol{s}$, and doing so does not (significantly) reduce the hardness of the problem.

Lemma 17 ([App+09])

There is a polynomial time reduction from short-secret-LWE$_{n,m,\chi,q}$ to LWE$_{n,m,\chi,q}$ and one from LWE$_{n,m,q,\chi}$ to short-secret-LWE$_{n,m+n,\chi,q}$

From SIS to LWE. We can see LWE as a "dual" problem to SIS. Let us start from SIS: we claimed before that, if the parameters are picked correctly, then SIS must have (at least) a solution. On the other hand, if we pick, e.g., $n < m \ll (n\log(q))/\log(\beta)$, then it is very unlikely that a solution $\boldsymbol{s}$ exists for $\boldsymbol{b}$ for a bounded $\|\boldsymbol{s}\| < \beta$. In other words:

$$\boldsymbol{b} = A^t\boldsymbol{z} \bmod q \tag{2.21}$$

for $A \sim \mathbb{Z}_q^{m\times n}$ and $m > n$ has no β-bounded solution, except $\boldsymbol{z}$, if the parameters are chosen in a certain way (e.g., $m \ll n\log(q)$). Notice that this corresponds precisely to an ISIS problem in the injective (low-density) setting, in particular we are in the case where $\beta < \lambda_1/2$.

By creating a point $\boldsymbol{b}$ that satisfies this equivalence – that is planting it – we are defining a coset (or a BDD target) in $\mathcal{L}_q(A)$. Nevertheless, in this case, recovering $\boldsymbol{s}$ is equivalent to solving LWE! We refer to this definition as the planted or LWE regime,

as we are *planting* the solution $\boldsymbol{b}$ to A, which is a lattice point that does not naturally belong to $\mathcal{L}_q(A)$. But how does this relate to LWE and (2.20)?

Consider an LWE instance $(A, \boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s})$, with $A \in \mathbb{Z}_q^{m \times n}$ uniformly sampled, and $m > n$, take for instance $m > n \log(q)$. In this setting, the left kernel of A has a large dimension, as we can efficiently build a matrix $A^\perp \in \mathbb{Z}_q^{(m-n)\times m}$ such that $A^\perp \cdot A = 0^{(m-n)\times m} \bmod q$.

If we then define $\boldsymbol{y} \in \mathbb{Z}_q^{m-n}$ as follows:

$$\boldsymbol{y} \equiv A^\perp \boldsymbol{b} \equiv A^\perp(A\boldsymbol{x} + \boldsymbol{s}) \equiv A^\perp \boldsymbol{s},$$

then $(A^\perp, \boldsymbol{y})$ is a inhomogeneous-SIS$_{m-n,m,q,\beta}$ instance whose solution is $\boldsymbol{z}$, which is short by definition. Also notice that the dimensions of A are $(m \times m - n)$, so we would need a much larger difference between $m - n$ and m to observe some possible solutions (except the planted one).

In particular, an SIS-solver can be used to distinguish LWE instances from uniformly random noise. The idea is as follows:

- Given an LWE instance $A \in \mathbb{Z}_q^{m \times n}$ and $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s} \in \mathbb{Z}_q^m$, we can build an SIS instance as $A^t \boldsymbol{z} \equiv \boldsymbol{0} \in \mathbb{Z}_q^m$.
- By solving the problem we find a short $\boldsymbol{z}$ and we compute

$$\begin{aligned} \boldsymbol{z}^t \boldsymbol{b} &\equiv \boldsymbol{z}^t(A\boldsymbol{x} + \boldsymbol{s}) \\ &\equiv \boldsymbol{z}^t A\boldsymbol{x} + \boldsymbol{z}^t \boldsymbol{s} \\ &\equiv (A^t \boldsymbol{z})^t \boldsymbol{x} + \boldsymbol{z}^t \boldsymbol{s} \equiv \boldsymbol{z}^t \boldsymbol{s} \bmod q \end{aligned} \tag{2.22}$$

- If $\boldsymbol{z}^t \boldsymbol{s}$ is small (since we know that both $\boldsymbol{z} \in \mathbb{Z}_q^m$ and $\boldsymbol{s} \in \mathbb{Z}_q^m$ are small), then $(A, \boldsymbol{b})$ was very likely an LWE instance, otherwise it was random noise.

So, we know that SIS is at least as hard as LWE. The other way round is also true, quantumly [Reg05]. It is an interesting open problem to see whether a classical reduction from LWE to SIS exists.

In order to better connect LWE and SIS, consider the involved parameters. In particular, we want the LWE error to be sampled with $\sigma > 0$, and we want it to be such that $\sigma \ll q$ otherwise the sampled $\boldsymbol{s}$ "wraps" around q. On average, the size $\|\boldsymbol{s}\|$ is set to be $\ll q$, so we set q/σ to be small but sufficiently large for the problem to be non trivial. For the SIS-solver to work correctly, we would need to find a vector $\boldsymbol{z}$ that, as $\boldsymbol{s}$

grows (respectively, σ), then the norm $\|\boldsymbol{z}\|$ (respectively β) becomes smaller. So LWE with σ, q, roughly corresponds to SIS with $\beta = q/\sigma$. The previous argument that shows how to distinguish the LWE distribution by looking at $\boldsymbol{z}^t\boldsymbol{s}$ is precisely an instantiation of this observation. Refer to Figure 49.

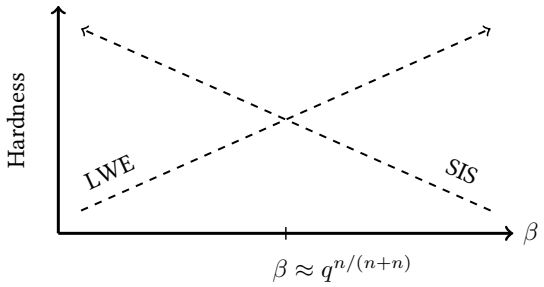


Figure 49: The hardness of LWE and SIS, for different β values. Taken from [Lyu24]. The lines represent the values such that the $\boldsymbol{z}^t\boldsymbol{s}$ product in Equation 2.22 stays small.

Algorithmically, we can see LWE and SIS as almost the same problem, although from a cryptographic point of view, they have many differences and unlock different opportunities. For instance, an open problem is about constructing a "decent" public-key cryptosystem from SIS, whereas with LWE the construction is immediate.

Remark that the planted regime point of view has the goal of "planting" a solution to the problem that does not exist. In particular, as we will see later, this solution can be used to artificially add a shortest vector (which corresponds to the unique SIS solution) to some random lattice, and it effectively works as a nice trapdoor as it is hard to recover but easy to solve if one knows the solution. We summarize some key points and differences between SIS and LWE in Table 2.

SIS	LWE
Search	Decisional
Many solutions $\boldsymbol{z}$	Unique solution $(\boldsymbol{x}, \boldsymbol{s})$
Surjective	Injective
LWE $\leq$ SIS (see above)	SIS $\geq$ LWE [Reg05]
Minicrypt	Cryptomania

Table 2: Comparison between SIS and LWE: two faces of the same coin. Taken from some slides by Peikert [Pei13]

We refer to [Imp95] for the definitions of Minicrypt and Cryptomania. By looking at LWE as a pair $(A, A\boldsymbol{x} + \boldsymbol{s})$, then we can look at it as an approximate BDD target in the q-ary lattice $\mathcal{L}_q(A)$. Remark that the latter is defined as shown in Definition 27, namely it is the set of solutions $\boldsymbol{y} \equiv A\boldsymbol{x}$. Also note that by definition it is contained in $\mathbb{Z}^m$, but the set of

[Imp95]: Impagliazzo (1995), *'A personal view of average-case complexity'* . In: Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference

points will be very sparse with respect to q especially when m is much larger than n, so the matrix is very tall. Indeed, the number of rows m should be larger than n but not *too* large, otherwise, it could be easier to invert the LWE function.

A lattice perspective. We saw (e.g. in Example 3) that SIS can be seen as an instance of approximate GapSVP over $\mathcal{L}_q^{\perp}(A)$. Learning with errors, on the other hand, can be seen as a "dual" problem, that is an approximate BDD over $\mathcal{L}_q(A)$. Recall the definition of $\mathcal{L}_q(A)$:

$$\mathcal{L}_q(A) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} \equiv A\boldsymbol{x} \text{ for some } \boldsymbol{x} \in \mathbb{Z}_q^n\}.$$

So the $\mathcal{L}_q(A)$ lattice is made of all the vectors $\boldsymbol{y} \in \mathbb{Z}^m$ for which there exists some $\boldsymbol{x} \in \mathbb{Z}_q^n$ such that $\boldsymbol{y} = A\boldsymbol{x}$. Basically, this corresponds to the set of LWE instances *without* errors.

An LWE instance, with $\boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s}$, can be seen as CVP target, with distance $\boldsymbol{s}$ from the lattice point $A\boldsymbol{x}$. It is possible to prove that the LWE lattice is not very dense, so the $\boldsymbol{b}$ target is not a lattice point with overwhelming probability. Moreover, $\boldsymbol{b}$ is guaranteed to be close to the lattice, therefore we can see $\boldsymbol{b}$ more appropriately as a BDD target. In the search version of LWE, one is asked to recover the closest vector $A\boldsymbol{x}$, while in the decision version one is asked whether $\boldsymbol{b}$ is a uniformly random vector or a noisy lattice point in $\mathcal{L}_q(A)$.

A knapsack perspective. At the beginning of this chapter, we presented a few ideas and results regarding knapsack-based cryptography. Then we claimed that lattices can be seen as knapsack's grandchild. To close the loop, consider an LWE instance – for simplicity with short secret – $(A, \boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s})$. We can look at this problem as the following knapsack:

$$\boldsymbol{b} = (A \mid I_m) \cdot \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{s} \end{pmatrix},$$

therefore given $\boldsymbol{b}$ and $(A \mid I_m)$, find a set of (small) coefficients such that $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s}$. The rows of $(A \mid I_m)$ are the weights of the knapsack, and the concatenation of $\boldsymbol{x}$ and $\boldsymbol{s}$ is the sequence that satisfies the equation.

The hardness of knapsacks is heavily related to their density, defined as the ratio between the input (weight) and the output (profit) lengths. In the LWE case, we have that the density is

roughly

$$\frac{(n+m)\log(\sigma)}{m\log(q)} \underbrace{\approx}_{\text{since } m>n} \frac{\log(\sigma)}{\log(q)} < 1$$

This also suggests that the larger the errors σ, the harder the problem. For a typical choice of parameters, e.g., $q = \text{poly}(n)$ and $\sigma \geq \sqrt{n}$, the density is within $(0, 1)$.

2.3.1 A simple public-key scheme from LWE

We will now give a simple description of a public-key encryption scheme based on LWE, called the Regev cryptosystem [Reg05]. The parameters are $m, n, q \in \mathbb{N}$, with $m > n\log(q)$. We first build a random basis for a q-ary lattice by uniformly sampling $A \in \mathbb{Z}_q^{m\times n}$. This corresponds to $\mathcal{L}_q(A)$. Then, we sample an LWE instance as $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s}$, with $\boldsymbol{x} \sim \mathbb{Z}_q^n$ and $\boldsymbol{s} \sim \chi^m$, for a discrete Gaussian distribution $\chi := \mathcal{D}_{\mathbb{Z},\sigma}$ with $\sigma = \sqrt{n}$. At this point, we build a new q-ary basis by appending $\boldsymbol{b}$ to A as a column as

[Reg05]: Regev, *'On lattices, learning with errors, random linear codes, and cryptography'* . In: STOC '05

$$pk = B := (\boldsymbol{b} \mid A) \in \mathbb{Z}_q^{m\times(n+1)}$$

Essentially, we are "planting" the short vector $\boldsymbol{s}$ to the $\mathcal{L}_q(A)$ lattice – this is why we use the $\boldsymbol{s}$ notation. Notice that, since we are in the "planted" regime, we are adding an exceptionally short to the lattice. The trapdoor is given by the secret key $sk := (1, -\boldsymbol{x}) \in \mathbb{Z}_q^{n+1}$, which allows one to recover the shortest vector $\boldsymbol{s} \in \mathcal{L}_q(B)$ as:

$$B \cdot \begin{pmatrix} 1 \\ -\boldsymbol{x} \end{pmatrix} \equiv (\boldsymbol{b} \mid A) \cdot \begin{pmatrix} 1 \\ -\boldsymbol{x} \end{pmatrix} \equiv \boldsymbol{b} - A\boldsymbol{x} \equiv \boldsymbol{s} \bmod q$$

So the public key is pk is the trapdoor q-ary basis B, and the secret key is $sk := (1 \mid -\boldsymbol{x})^t \in \mathbb{Z}_q^{n+1}$. From a lattice perspective, the public key is a bad lattice basis with a planted short vector, and the secret key is a way to recover its shortest vector. To encrypt a message μ, without loss of generality a binary value, we take advantage of some dual lattice properties. Remark that (Figure 36) having access to the shortest dual vector, one can partition the primal lattice, and vice versa. In our setting, we have access to the shortest primal vector, so we sample an encryption as a point in the dual lattice $\mathcal{L}_q^\perp(B)$. By evaluating

$$\boldsymbol{e} := B^t\boldsymbol{r} \in \mathbb{Z}_q^n$$

for some binary $\boldsymbol{r} \in \{0,1\}^m$, we are defining a coset $\boldsymbol{e}+\mathcal{L}_q^\perp(B)$, or a BDD target in $\mathcal{L}_q^\perp(B)$. Recovering $\boldsymbol{r}$, given $\boldsymbol{e}$, implies a solution to ISIS, which we know is hard. So we can use $\boldsymbol{e} \in \mathbb{Z}_q^n$ as a pseudorandom mask for some message. A ciphertext can thus be sampled as

$$\boldsymbol{c} := \boldsymbol{e} + (q/2 \cdot \mu, 0, \dots, 0) \in \mathbb{Z}_q^n$$

We can decode the ciphertext by evaluating an inner product between the small error $\boldsymbol{r}$ and $\boldsymbol{s}$: if the result is close to zero, then the target is close to the origin, thus, an encryption of zero. On the other hand, if the result is close to $q/2$, then the message is one. Decryption is therefore simply evaluated as:

$$\begin{aligned}[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q &\equiv [\langle B^t \boldsymbol{r} + (q/2 \cdot \mu, 0, \dots, 0), (1 \mid -\boldsymbol{x})^t \rangle]_q \\ &\equiv [\langle \boldsymbol{s}, \boldsymbol{r} + (q/2\mu, 0, \dots, 0) \rangle]_q\end{aligned}$$

and is correct as long as the amount of noise $\langle \boldsymbol{r}, \boldsymbol{s} \rangle$ is smaller than $q/4$. Indeed, it is clear that if $[\langle \boldsymbol{r}, \boldsymbol{s} \rangle]_q$ is larger than $q/4$, then the result wraps around q and becomes incorrect (e.g., an encryption of zero with noise slightly larger than $q/4$ becomes larger to $q/2$ than the origin, so gets decrypted to one). We give an intuition on such process in Figure 50: the yellow points (close to the partitions) are encryptions of zero, while the orange ones are encryptions of one. Of course the space

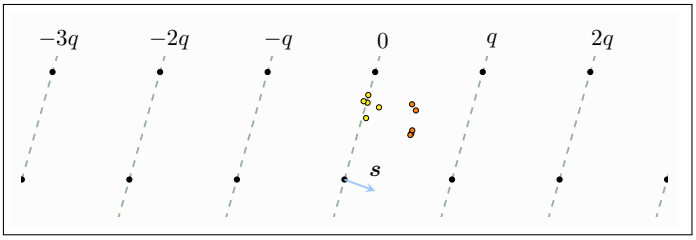


Figure 50: Encryptions in the dual lattice $\mathcal{L}_q^\perp(A)$. Every partition i is the set of points that have inner product $i \cdot q$ with the shortest primal vector $\boldsymbol{s} \in \mathcal{L}_q(A)$

can be discretized even more to allow for the encryption of larger integer values.

Correctness. We know prove the correctness of this simple scheme by probabilistic arguments.

Lemma 18 (*Correctness*)

By setting $\chi = \mathcal{D}_{\mathbb{Z},r}$ for some $r > 0$ and $q > r\sqrt{m+1}\log(n)$, the just presented scheme is correct with superpolynomial probability.

Proof. Remark that the relation between the public and the secret key is such that

$$B \cdot \begin{pmatrix} 1 \\ -\boldsymbol{x} \end{pmatrix} \equiv \boldsymbol{s} \bmod q,$$

and the decryption relies on the norm of $\boldsymbol{s}$ since:

$$[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q \equiv [\langle \boldsymbol{r} + (q/2\mu, 0, \ldots, 0), \boldsymbol{s} \rangle]_q$$

Remark that $s_1 \equiv 1$, so it suffices to show that $\langle \boldsymbol{r}, \boldsymbol{s} \rangle \in \mathbb{Z}$ has magnitude less than $\lfloor q/4 \rfloor$ with high probability (conversely, the parity of μ gets changed). We have that, by definition $\|\boldsymbol{r}\| \leq \sqrt{m+1}$ and $\boldsymbol{s} \in \chi^m$. By tail bounds on $\mathcal{D}_{\mathbb{Z},r}$, we have that $\|\boldsymbol{s}\| < tr$ except with probability at most $2e^{-\pi t^2}$ for some $t > 0$. By Cauchy-Schwarz we have that

$$|\langle \boldsymbol{r}, \boldsymbol{s} \rangle| \leq \|\boldsymbol{r}\| \cdot \|\boldsymbol{s}\| \leq \sqrt{m+1} \cdot rt.$$

The result follows by setting $t = \log(n)$. □

2.4 Algebraic lattices

In cryptography security is important, but efficiency is a crucial aspect as well, especially when deploying schemes to a large scale. The main issue when considering lattices in cryptography, is about their description. A lattice public basis is quadratic with respect to the dimension n, meaning that for sufficiently large dimensions, its size becomes unpractical. In the case of LWE, usually the public key has a shape $m \times n$, where $m = \mathcal{O}(n \log(q))$. An example of a public-key construction from plain LWE is FrodoKEM [Bos+16], one of the NIST candidates in the post-quantum cryptography standardization process, eventually not selected because of its inefficiency. This naturally leads to the question:

[Bos+16]: J. Bos, Costello, Ducas, Mironov, et al., *'Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE'* . In: CCS '16

Can we construct lattices that allow for short descriptions?

The answer is yes, and we call these particular lattices *algebraic lattices* (or *structured lattices*). Note: the following definitions and concepts are not meant to be exhaustive, but only to introduce some fundamental ideas required later. For a more thorough explanation and proofs, we refer to the following manuals [Ste04; Mar18] and to the original papers.

[Ste04]: Stein (2004), A Brief Introduction to Classical and Adelic Algebraic Number Theory
[Mar18]: Marcus (2018), Number Fields

2.4.1 Structured lattices

Originally inspired by the NTRU cryptosystem [HPS98], the reason why structured lattices are useful is that they allow for succinct representations, as it is possible to define an n-dimensional lattice with a single vector. Moreover, because of the algebraic structure of the underlying lattice, it is possible to accelerate (e.g. via FFT) some procedures, and finally they make hardness proofs possible. One of the first examples is a one-way function based on worst-case hardness of SVP in cyclic lattices [Mic02].

[HPS98]: Hoffstein, Pipher, and Silverman (1998), *'NTRU: A ring-based public key cryptosystem'* . In: Algorithmic Number Theory

[Mic02]: Micciancio, *'Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions'* . In: FOCS '02

2.4.1.1 Cyclic lattices

Consider a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, then define the cyclic rotation $\text{rot}(\boldsymbol{x}) = (x_n, x_1, \ldots, x_{n-1})$ and the corresponding circulant matrix:

$$\text{Rot}(\boldsymbol{x}) = (\boldsymbol{x}, \text{rot}(\boldsymbol{x}), \text{rot}^2(\boldsymbol{x}), \ldots, \text{rot}^{n-1}(\boldsymbol{x})) \tag{2.23}$$

Definition 33 (*Cyclic lattice*)

A lattice $\mathcal{L}$ is said to be cyclic *if it is closed under rotation, namely for any $\boldsymbol{x} \in \mathcal{L}$, then* $\text{rot}(\boldsymbol{x}) \in \mathcal{L}$.

Notice that this class of lattices allows for a short description as we are only required to store the n elements of some $\boldsymbol{b}$ vector to refer to a full dimensional lattice (obtained by simply computing all the rotations of $\boldsymbol{b}$). These lattices have been used for, e.g., the construction of efficient one-way functions based on the hardness of solving lattice problems in the worst case over cyclic lattices [Mic02]. A few years later, Peikert and Rosen [PR06] and Lyubashevsky and Micciancio [LM06] concurrently showed that such one-way functions can also be made collision-resistant[16]. Intuitively, given $f_A(\boldsymbol{x}) : \{0,1\}^m \to \mathbb{Z}_q^n$, with $m < n \log(q)$, a hashed message corresponds to a lattice point in the q-ary lattice $\mathcal{L}^\perp(A)$. Finding a collision means to find $\boldsymbol{x}' \in \{0,1\}^m$ such that $f_A(\boldsymbol{x}) = f_A(\boldsymbol{x}')$, but if this holds, then we have $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}' \in \{-1, 0, 1\}^n$ such that $f_A(\boldsymbol{z}) = f_A(\boldsymbol{x}) - f_A(\boldsymbol{x}') = 0$, thus we found a solution to SIS.

[PR06]: Peikert and Rosen, *'Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices'* . In: TCC'06

[LM06]: Lyubashevsky and Micciancio, *'Generalized compact knapsacks are collision resistant'* . In: ICALP'06

16: Other than being one-way, it is also very hard to find two inputs a, b such that $f(a) = f(b)$

2.4.1.2 Ideal lattices

We now introduce the more general class of *ideal lattices*, that is lattices generated by *ideals*. One of the main foundations of lattice-based cryptography, and its efficiency, is about the relationship between some families of lattices, the algebra of polynomials modulo $X^n + 1$ and $X^n - 1$, and linear algebra in $\mathbb{R}^n$.

Let us start by introducing the concept of ideal, which will be essential for our next constructions.

Definition 34 (*Ideal*)

Given a ring $\mathcal{R}$, an ideal $\mathcal{I}$ is a subgroup $\mathcal{I} \subseteq \mathcal{R}$ such that for every $r \in \mathcal{R}$ and $i \in \mathcal{I}$, the product $r \cdot i \in \mathcal{I}$.

Intuitively, think of an ideal as a subgroup of a ring where the multiplication between any element of the ideal and any element of the ring "absorbes" the result into the ideal[17].

17: For example, the set of even values in $\mathbb{Z}$ is an ideal since the multiplication between any value and a even value is even

Definition 35 (*Polynomial ring*)

The ring $\mathbb{Z}[X]$, defined as

$$\mathbb{Z}[X] = \{a_0 + a_1 X + a_2 X^2 + \cdots + a_n X^n : n \geq 0, a_i \in \mathbb{Z}\}$$

is called polynomial ring *over the integers.*

Simply, $\mathbb{Z}[X]$ defines a ring which contains all the possible polynomials with integer coefficients. As a next step, we introduce quotient rings.

Definition 36 (*Quotient ring*)

Given a ring $\mathcal{R}$ and an ideal $\mathcal{I}$ of $\mathcal{R}$, the quotient ring $\mathcal{R}/\mathcal{I}$ *is defined as*

$$\mathcal{R}/\mathcal{I} = \{r + \mathcal{I} : r \in \mathcal{R}\}.$$

As an example, let us consider the $\mathbb{Z}$ ring. A natural ideal is $2\mathbb{Z}$, the set of even integers, so we can build the quotient ring $\mathbb{Z}/2\mathbb{Z}$, commonly referred to as $\mathbb{Z}_2$, which gives a structure made of two elements only. In this quotient ring, elements in $\mathbb{Z}$ that are equal to elements in $2\mathbb{Z}$ are set to 0, so it contains two elements only:

$$\mathbb{Z}/2\mathbb{Z} = \{0 + 2\mathbb{Z}, 1 + 2\mathbb{Z}\}$$

We will be interested in a specific type of ideals, called *principal ideals*.

Table 3: Summary of the coefficients embedding

	Polynomials	Lattice
Representation	$a_0 + a_1X + \ldots, a_{n-1}X^{n-1}$	Lattice point $(a_0, a_1, \ldots, a_{n-1})$
Addition	$a(X) + b(X) \bmod p(X)$	$\boldsymbol{a} + \boldsymbol{b}$
Multiplication	$a(X) \cdot b(X) \bmod p(X)$	(Not defined)

Definition 37 (*Principal ideal*)

Given a ring $\mathcal{R}$, let $p \in \mathcal{R}$. The set

$$\langle p \rangle = \{p \cdot r : r \in \mathcal{R}\}$$

is the principal ideal *generated by p.*

Notice that such ideals consist of all the elements in $\mathcal{R}$ multiplied by some ring element p. For example, the previously considered ideal $2\mathbb{Z}$ is principal, as it is generated by multiplying each element in $\mathbb{Z}$ by 2.

We can therefore consider the more general polynomial ring $\mathbb{Z}[X]$ and fix some quotient ring of it by picking any polynomial $p(x) \in \mathbb{Z}[X]$

$$\mathbb{Z}[X]/\langle p(X) \rangle$$

where $p(X)$ is some monic[18] n-dimensional polynomial in $\mathbb{Z}[X]$. It turns out that this quotient ring forms a lattice and in order to better understand this connection, simply consider the *coefficients embedding* map ϕ:

18: Such that the leading coefficient $a_n = 1$

$$\begin{aligned} \phi : \mathbb{Z}^n &\to \mathbb{Z}[X]/\langle p(X) \rangle, \\ (v_0, v_1, \ldots, v_n) &\to v_0 + v_1X + \cdots + v_nX^{n-1} \end{aligned} \tag{2.24}$$

which induces a group isomorphism by considering the coefficients of the polynomials as the lattice points of $\mathbb{Z}^n$. Notice that this isomorphism holds as a $\mathbb{Z}$-module (so as an additive group), and not as a ring, since there is no operation between lattice vectors that reflects convolution between two polynomials in $\mathbb{Z}[X]/\langle p(X) \rangle$. Later we will introduce a different embedding that will preserve also multiplications. We give an overview of the coefficients embedding in Table 3.

We also give a brief example on how to interpret a quotient ring (Example 4).

Example 4 (*Quotient ring to lattice*)

Given the (integer) polynomial ring $\mathbb{Z}[X]$, fix a polynomial $p(X) = X^3$. We can define a principal ideal formed by all

the multiples of $p(X)$, therefore we can also define the quotient ring $\mathbb{Z}[X]/\langle X^3\rangle$, which is formed by all the classes of polynomials modulo $X^3 = 0$, so every element can be written as

$$a + bX + cX^2 \text{ with } a, b, c \in \mathbb{Z}$$

Now simply use the ϕ map to obtain all the possible $(a, b, c) \in \mathbb{Z}^3$, and observe that $\mathbb{Z}^3$ is indeed formed by those vectors.

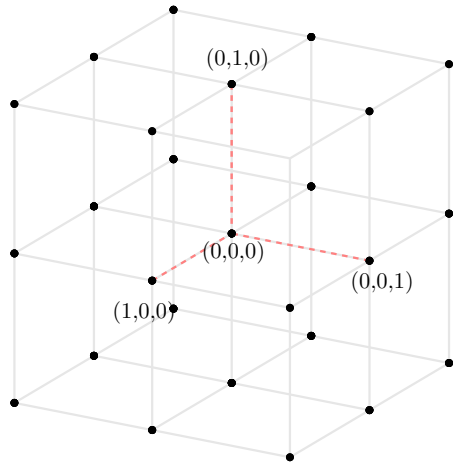


Figure 51: (Part of) $\mathbb{Z}^3$ generated by the coefficients embedding of all the elements from $\mathbb{Z}[X]/\langle X^3\rangle$

In our interpretation, every lattice point corresponds to a polynomial of degree at most 2, and the coordinates (a, b, c) of the point are the coefficients of the polynomial.

The example implicitly explains why we require monic polynomials when constructing lattices. By picking, e.g., $p(X) = 2X^3 + 1$, we would obtain a "shrink" of the third axis since $X^3 = -\frac{1}{2}$, leading to something different from $\mathbb{Z}^3$.

Notice that we could plug any monic polynomial $p(X)$ of degree n in to obtain a n-dimensional lattice under this embedding. Nevertheless, we will be later interested in particular polynomials because of some interesting properties. For example, by picking $p(x) = X^2$, one has that $X^2 = 0$, so multiplications by X degenerate to zero. On the other hand, by picking, e.g., $p(X) = X^2 + 1$, one has that $X^2 = -1$, so multiplicating an element of the ring by X "rotates" it as a complex number[19]. So we are always able to generate a lattice as an additive group, but the choice of $p(X)$ changes the "geometry under multiplication". In particular, in order to en-

19: Observe the similarity between $X^2 = -1$ and $i^2 = -1$

rich the structure, we will be interested in using polynomials which are also irreducibile[20]. This property also ensures that the modulus can not be split in smaller ones and prevents some attacks – see, e.g., Section 4.1 of [PR06].

20: With the term *irreducible*, we mean irreducible over the rationals

Usually, in lattice cryptography, we work with sublattices of $\mathbb{Z}^n$ – so that the underlying problems become nontrivial. A natural way to construct a sublattice $\mathcal{L} \subseteq \mathbb{Z}^n$ is to consider some matrix $B \in \mathbb{Z}^{n \times n}$ as a basis to obtain $\mathcal{L}(B) \subset \mathbb{Z}^n$.

Equivalently, in polynomials terms, we can consider a subring of $\mathbb{Z}^n$, therefore by picking a second ideal. Not to get confused, $p(X)$ generates an ideal in $\mathbb{Z}[X]$ that corresponds to $\mathbb{Z}^n$. The second ideal that we use is an ideal of $\mathbb{Z}[X]/\langle p(X)\rangle$, which can, as a principal ideal, be any polynomial of degree less than n, for instance $\langle g(X)\rangle \subseteq \mathbb{Z}[X]/\langle p(X)\rangle$ – up to equivalence mod $p(X)$. We thus define the ideal $\mathcal{I}$ as

$$\mathcal{I} = \{g(X) \cdot h(X) \bmod p(X) : h(X) \in \mathbb{Z}[X]/\langle p(X)\rangle\},$$

which will represent a sublattice of $\mathbb{Z}^n$ via the usual mapping ϕ. Notice that we might sometime use the following, more compact, notation:

$$\mathcal{I} = \mathbb{Z}[X]/\langle p(X), g(X)\rangle.$$

In particular, the family of lattices generated by this kind of ideals is referred to as *ideal lattices*.

Definition 38 (*Ideal lattice*)

An ideal lattice is the set of coefficient vectors corresponding to the elements of an ideal $\mathcal{I} \subseteq \mathbb{Z}[X]/\langle p(X)\rangle$, for some monic, irreducible $p(X)$ of degree n. We denote

$$\mathcal{L}(\mathcal{I}) = \{(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n : a(X) \in \mathcal{I}\}$$

where $a(X) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$.

To keep things more clear, we report the objects used until now, with their respective meaning, in Table 4.

From	To	Ideal	Geometry
$\mathbb{Z}[X]$	$\mathcal{R} = \mathbb{Z}[X]/\langle p(X)\rangle$	$p(X) \in \mathbb{Z}[X]$	$\mathbb{Z}^n$
$\mathcal{R}$	$\mathcal{I} = \mathcal{R}/\langle h(X)\rangle \subseteq \mathcal{R}$	$h(X) \in \mathcal{R}$	Subset of $\mathbb{Z}^n$

Table 4: Summary of $\mathbb{Z}^n$ and polynomial relationships

We remark that every ideal of $\mathcal{R}$ induces a sublattice $\subseteq \mathbb{Z}^n$, but not every sublattice has a corresponding ideal. Indeed, the domain of all possible full dimensional lattice bases is

necessarily larger than the ones that can be defined with a polynomial in space $\mathcal{O}(n)$ (Pigeonhole principle).

We can summarize these ideas in the following.

Lemma 19 ([LM06, Lemma 3.2])

Every ideal $\mathcal{I}$ of $\mathbb{Z}[X]/\langle p(X)\rangle$, where $p(X)$ is a monic, irreducible integer polynomial of degree n, is isomorphic to a full dimensional lattice in $\mathbb{Z}^n$.

To summarize, we also give an example on how to define a cyclic lattice, starting from a polynomial (Example 5), exploiting the polynomial quotient defined by the $X^n - 1$ modulus.

Example 5 (*Polynomial → Cyclic lattice*)

Consider the ring

$$\mathcal{R} = \mathbb{Z}[X]/\langle X^4 - 1\rangle$$

which defines the $\mathbb{Z}^4$ lattice using the isomorphism on the coefficients (2.24). Remark that every polynomial in $\mathcal{R}$ has the form

$$a_0 + a_1 X + a_2 X^2 + a_3 X^3$$

with $(a_0, a_1, a_2, a_3) \in \mathbb{Z}^4$, can be seen as a lattice point. We can now fix some (random) polynomial $h(X)$ that will be our principal ideal as, for instance

$$h(X) = 3 + X + 2X^2 - 1X^3.$$

So, the principal ideal $\mathcal{I}$ generated by $h(X)$ is the set

$$\mathcal{I} = \{h(X) \cdot p(X) \bmod (X^4 + 1) : p(X) \in \mathcal{R}\}$$

which in turn defines our (sub)lattice $\mathcal{L} \subset \mathbb{Z}^4$ via the coefficient embedding. Now, if we want to find an actual basis $B \in \mathbb{Z}^4$ such that $\mathcal{L} = B \cdot \mathbb{Z}^4$, we can consider the coefficients of $h(X)$ as the first basis vector:

$$\boldsymbol{b}_1 = (3, 1, 2, -1)$$

then, in order to find the other basis vectors, we simply multiply $h(X)$ with all the powers of X. For example, to get $\boldsymbol{b}_2$:

$$h(X) \cdot X = 3X + X^2 + 2X^3 - X^4$$

we reduce mod $X^4 - 1$ by observing that $X^4 = 1$, to obtain

$$h(x) \cdot x = -1 + 3X + X^2 + 2X^3$$

which gives the second basis vector

$$\boldsymbol{b}_2 = (-1, 3, 1, 2).$$

Notice that this vector is precisely $\text{rot}(\boldsymbol{b}_1)$, and it is easy to see that the resulting basis vectors form a circulant matrix! We therefore obtain the following lattice basis

$$B = \begin{pmatrix} 3 & -1 & 2 & 1 \\ 1 & 3 & -1 & 2 \\ 2 & 1 & 3 & -1 \\ -1 & 2 & 1 & 3 \end{pmatrix} \in \mathbb{Z}^{4\times 4},$$

which indeed defines a full dimensional lattice that lives in the larger $\mathbb{Z}^4$ space.

As it will be shown later for the ring version of SIS or LWE, the choice of the modulus polynomial is usually set to a *cyclotomic polynomial* – that will be defined later – of the form $\Phi_n(X) = X^n + 1$, with n being a power of two. Notice that, differently from the example where $X^4 = -1$, the rotations acts slightly differently as the rotated values will also have their sign changed.

The class of *cyclotomic rings* in the form $\mathbb{Z}[X]/\Phi_n(X)$, where $\Phi_n(X)$ is the n-th cyclotomic polynomial (which has degree $\varphi(n)$, is monic and irreducible) is very popular in cryptography it has some nice properties. For instance, as it will be better explained in Section 2.4.3, by using such class of polynomials we can prove that the search and the decision versions of the ring version of LWE are equivalent (up to a polynomial factor) and also we end up in a win-win situation, as power of two polynomials can be implemented very efficiently.

2.4.1.3 A number fields perspective

We now introduce some concepts from algebraic number theory that will provide a different perspective on constructing lattices, along with an important embedding that allows us to define a way to construct lattices in $\mathbb{R}^n$ from a so-called *ring of integers*, preserving additions and multiplications.

Definition 39 (*Algebraic number*)

An algebraic number *is a number that is a root of a nonzero polynomial with rational coefficients.*

Definition 40 (*Number field*)

A number field $\mathbb{Q}(x)$ *is a finite-degree extension of* $\mathbb{Q}$ *with* x *being an algebraic number.*

With *degree* we refer to the dimension of the field as a vector space in $\mathbb{Q}$. Intuitively, we pick the field $\mathbb{Q}$ and we extend it with some element; in particular, an *algebraic number*.

For example, $\sqrt{2}$ is an algebraic number since it is the root of $X^2 - 2$. Following the example, we can use $\sqrt{2}$ to augment $\mathbb{Q}$, namely $\mathbb{Q}(\sqrt{2})$. This number field refers to all the values in the form $a + \sqrt{2}b$, where $a, b \in \mathbb{Q}$.

Now, for our constructions, we are particularly interested in number fields constructed using the *roots of unity*.

Definition 41 (*n-th roots of unity*)

An n*-th* root of unity *is a number* $x \in \mathbb{C}$ *such that* $x^n = 1$.

In general, we consider the *primitive* n-th root of unity[21] $\xi_n = e^{2\pi i/n}$. From it we can indeed derive all the others as $1, \xi, \xi^2, \ldots, \xi^{n-1}$.

21: Sometimes the notation for roots of unity is ω

When $n = 1$, this is trivial as the only number is $x = 1 + 0i$. When $n = 2$, then we would have to solve $x^2 = 1$, leading to the two solutions 1 and -1; notice that $e^{\pi i} = -1$ by Euler's identity. It gets interesting starting from $n = 3$ since we would have to solve $x^3 = 1$, and we find three different solutions to this equation:

- $x = 1$, trivial.
- $x = e^{2\pi i/3}$, since $(e^{2\pi i/3})^3 = e^{6\pi i/3} = e^{2\pi i} = 1$.
- $x = e^{4\pi i/3}$, since $(e^{4\pi i/3})^3 = e^{12\pi i/3} = e^{4\pi i} = 1$.

Note that we can conveniently draw these solutions over the complex plane (as shown in Figure 52). Furthermore, it is easy to draw them as they are evenly spaced along the unit circle[22].

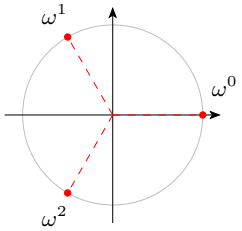


Figure 52: The $n = 3$ roots of unity

Also remark that, as these roots are actually the solutions of $x^n - 1 = 0$ in $\mathbb{C}$, the following theorem must hold.

22: This happens because each root is obtained by squaring the previous one, leading to every root simply being the previous one, rotated by $2\pi/n$

Theorem 6 (*Conjugate root theorem*)

Given a polynomial p *with real coefficients with* $a + bi \in \mathbb{C}$ *being one of its roots, then its complex conjugate* $a - bi$ *is also a root for* p.

This nicely translates in the fact that roots of unity are symmetric with respect to the $\mathbb{R}$ axis. We now close the loop by connecting roots of unity and number fields, obtaining *cyclotomic number fields* and *cyclotomic polynomials*.

Definition 42 (*n-th cyclotomic number field*)

*The n-*th cyclotomic number field *is the number field $\mathbb{Q}[\xi_n]$, where $\xi_n = e^{2\pi i/n}$ is the n-th primitive root of unity.*

We can think of $\mathbb{Q}[\xi_n]$ as a field containing all the rational numbers, all the n-th roots of unity and all rational combinations of powers of ξ_n.

Definition 43 (*n-th cyclotomic polynomial*)

*The n-*th cyclotomic polynomial *$\Phi_n(X)$ is the unique irreducible polynomial with integer coefficients that is a divisor of $X^n - 1$ and whose roots are all the n-th roots of unity:*

$$\Phi_n(x) = \prod_{1 \leq k \leq n, \gcd(k,n)=1} (x - e^{(2\pi k i)/n})$$

Until now, we used ideals of quotient rings to construct lattices. This perspective can be somewhat limited because of the coefficients embedding that we used above. It turns out that another perspective is to use number fields, and this is indeed the point of view usually adopted to perform hardness proofs. This allows us to introduce another embedding, called *Minkowski embedding*[23] (or *canonical embedding*).

23: Notice that the difference between embeddings is up to a linear transformation in $\mathbb{R}^n$

Let us start from $\mathbb{Q}[X]$, that is, the ring of polynomials with rational coefficients. Now, fix some irreducible polynomial $p(X) \in \mathbb{Q}[X]$ of degree n, and consider the quotient ring

$$\mathbb{Q}[X]/\langle p(X) \rangle = \{g(X) + p(X) : g(X) \in \mathbb{Q}[X]\}$$

Observe that, since $p(X)$ is irreducible, the ideal $\langle p(X) \rangle$ is *maximal*[24], and we are forming a field.

24: There is no other ideal $\mathcal{I}_2$ in which it is contained, i.e., $\mathcal{I} \not\subset \mathcal{I}_2$

Lemma 20

The quotient ring $\mathcal{R}/\mathcal{I}$, with $\mathcal{I}$ being a maximal ideal, is a field.

Proof. Let $\mathcal{I}$ be a maximal ideal of $\mathcal{R}$. For any $r \notin \mathcal{I}$, the ideal $\mathcal{I} + (r) = \mathcal{R}$, so $1 \in \mathcal{I} + (r)$. Hence, there exists $s \in \mathcal{R}$ such that

$$rs \equiv 1 \bmod \mathcal{I}.$$

Thus every nonzero coset $r + \mathcal{I}$ is invertible in $\mathcal{R}/\mathcal{I}$, so $\mathcal{R}/\mathcal{I}$ is a field. □

In particular, we are generating a number field. Remark that $p(X)$ is irreducible, meaning that we can not find solutions[25] in $\mathbb{Q}$. By generating that quotient ring, we are forcing the $p(X) = 0$ condition, meaning that we are adding the

$$\alpha = X \bmod p(X)$$

element to $\mathbb{Q}$, that is precisely the way of constructing a number field, namely we are adding a solution of some irreducible f to $\mathbb{Q}$. Every element of this number field is in the form

$$a_0 + a_1\alpha + a_2\alpha^2 + \dots a_{n-1}\alpha^{n-1}, \quad a_i \in \mathbb{Q},$$

so the two objects $\mathbb{Q}[X]/\langle p(X)\rangle$ and $\mathbb{Q}(\alpha)$ are the same. In fact, it is well known that

$$\mathbb{Q}[X]/\langle f\rangle \cong \mathbb{Q}(\alpha) \qquad \text{for } \alpha \text{ such that } f(\alpha) = 0.$$

25: If there were, we could factor $p(x)$ as $p(x) = (x - \alpha)g(x)$, for a root $\alpha \in \mathbb{Q}$ and some $g(x) \in \mathbb{Q}[X]$, but this is not possible by definition

Definition 44 (*Minkowski embedding*)

Let $K = \mathbb{Q}[X]/\langle p(X)\rangle$ be a number field for some irreducible $p(X) \in \mathbb{Q}[X]$. This gives deg*(p) different field embeddings into $\mathbb{C}$, each given by evaluation at a root α_i of $p(X)$*

$$\sigma_i : X \mapsto \alpha_i$$

This embedding is called Minkowski embedding *(sometimes referred to as* canonical embedding*).*

The main difference with the previous naïve coefficients embedding ϕ, is that before each element of the quotient ring was written as a linear combination of $(1, x, \dots, x^{n-1})$, while under σ every element is written as a linear combination of $(1, \alpha, \dots, \alpha^{n-1})$. Notice that until now we worked with the number field $K = \mathbb{Q}[X]/\langle p(X)\rangle$, although we will focus more on the ring of integers $\mathcal{O}_k$

Definition 45 (*Ring of integers*)

The ring of integers $\mathcal{O}_K$ *of a number field K is the ring of all the algebraic numbers in K.*

We will sometimes refer to the ring of integers also as *number ring*. The image of the canonical embedding over a ring of integers is a lattice. Let us briefly consider a quick example.

Example 6 (*Ring of integers $\xrightarrow{\sigma}$ lattice*)

Let us define the number field $K = \mathbb{Q}(\sqrt{2})$. The latter is

defined as the set of all numbers in the form

$$K = \{a + b\sqrt{2} : a, b \in \mathbb{Q}\}$$

Remark that the ring of integers $\mathcal{O}_K \subset K$ is the set of all algebraic numbers in K, meaning all the roots of polynomials with integers coefficients in K, thus in the form

$$\mathcal{O}_K = \{a + \sqrt{2}b, \quad a, b \in \mathbb{Z}\}$$

Now, consider the Minkoswki embedding σ, that allows one to map the ring of integers to a lattice. The degree of the number ring is 2 (since $\sqrt{2}$ is the root of a degree 2 polynomial $x^2 - 2$), meaning that we have 2 embeddings, one for each root σ_i of the minimal polynomial of $\sqrt{2}$ over $\mathbb{Q}$, namely $x^2 - 2$:

$$\sigma_1 : \sqrt{2} \to \sqrt{2}$$
$$\sigma_2 : \sqrt{2} \to -\sqrt{2}$$

meaning that (by abuse of notation)

$$\sigma_1 \leftarrow a + \sqrt{2}b$$
$$\sigma_2 \leftarrow a - \sqrt{2}b$$

so, the embedded set becomes

$$\{(a + \sqrt{2}b, a - \sqrt{2}b) : a, b \in \mathbb{Z}\}.$$

Now, as a last step, notice that we can look at the set as

$$a(1, 1) + b(\sqrt{2}, -\sqrt{2}),$$

giving us the two basis vectors

$$\boldsymbol{b}_1 = (1, 1)$$
$$\boldsymbol{b}_2 = (\sqrt{2}, -\sqrt{2})$$

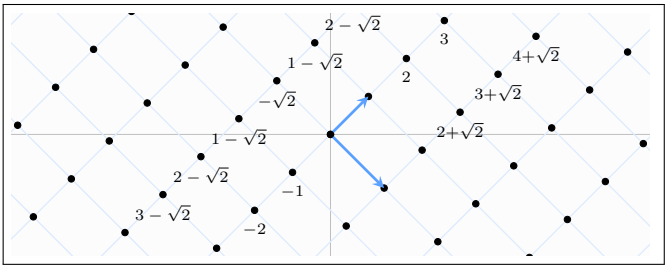


Figure 53: The lattice generated by the Minkoswki embedding on $\mathbb{Q}(\sqrt{2})$

So, the process to obtain a lattice out of a number field is

$$\mathbb{Q}[X] \xrightarrow{\text{Contains}} \mathcal{O}_K \xrightarrow{\text{Canonical embedding}} \mathcal{L}$$

Another way to see this is via the Vandermonde matrix, defined as

$$V = \begin{pmatrix} 1 & \alpha_1 & \dots & (\alpha_1)^{n-1} \\ 1 & \alpha_2 & \dots & (\alpha_2)^{n-1} \\ 1 & \alpha_3 & \dots & (\alpha_3)^{n-1} \\ 1 & \alpha_4 & \dots & (\alpha_4)^{n-1} \end{pmatrix} \in \mathbb{C}^{4\times 4},$$

where α_i is the i-th root of the minimal degree n polynomial that defines the number field. In the case of $K = \mathbb{Q}[\sqrt{2}]$, we saw that the roots are $\sqrt{2}$ and $-\sqrt{2}$, resulting in the following matrix:

$$V = \begin{pmatrix} 1 & \sqrt{2} \\ 1 & -\sqrt{2} \end{pmatrix} \in \mathbb{C}^{2\times 2},$$

which is the basis for the lattice obtained by considering $\mathcal{O}_K$.

Remark 3

Under the coefficients embedding ϕ, we can define a lattice simply by picking the set of coefficients. Under the Minkoswki embedding, we define the lattice basis starting from the Vandermonde matrix V. Observe that V is invertible and has rank n, so it generates the whole $\mathbb{Z}^n$ space.

Notice that, differently from the previous coefficients embedding, which gives rise to a group homomorphism with respect to additions, this embedding defines a ring homomorphism, as:

$$\begin{aligned} \sigma(a+b) &= \sigma(a) + \sigma(b), \\ \sigma(a \cdot b) &= \sigma(a) \odot \sigma(b) \end{aligned} \tag{2.25}$$

where $a, b \in \mathcal{O}_K$ and $\odot$ indicates the coordinate-wise multiplication. On the other hand, the coefficients embedding ϕ does not preserve the multiplication. Nevertheless, these two embeddings differ by a linear transformation in $\mathbb{C}^n$, as it will be shown in Example 8[26].

Example 7 (*Ring homomorphisms*)

26: The transformation between these two representations is crucial, especially for encoding/decoding and for bootstrapping operation in FHE schemes

Consider the ring $\mathbb{Z}[X]/(X^2+1)$ and two polynomials $a(X) = 2 + 4X, b(X) = 1 - 3X$. We want to show how addition and multiplication work under the Minkowski embedding. Let us compute the two operations:

$$a(X) + b(X) = 2 + 4X + 1 - 3X = 3 + X$$
$$\sigma(a(X) + b(X)) = (3 + i, 3 - i),$$

and

$$a(X) \cdot b(X) = 2 - 6X + 4X + 12 = 14 - 2X$$
$$\sigma(a(X) \cdot b(X)) = (14 - 2i, 14 + 2i).$$

Now let us compute their embeddings, and perform the operations there:

$$\sigma(a(X)) = (\sigma_1(a(X)), \sigma_2(a(X))) = (2 + 4i, 2 - 4i)$$
$$\sigma(b(X)) = (\sigma_1(b(X)), \sigma_2(b(X))) = (1 - 3i, 1 + 3i)$$

Addition:

$$\sigma(a(X)) + \sigma(b(X)) = (3 + i, 3 - i).$$

Point-wise multiplication:

$$\sigma(a(X)) \odot \sigma(b(X)) = ((2 + 4i) \cdot (1 - 3i), (2 - 4i) \cdot (1 + 3i))$$
$$= (14 - 2i, 14 + 2i)$$

To summarize, this embedding defines a lattice by evaluating the polynomial over all its roots, and by doing so it is possible to use classic results in number theory to define a nice geometric setting, since by using σ we know that the norm

$$\|a\| = \|\sigma(a)\|$$

This property is crucial for hardness proofs.

Cyclotomic polynomials. In actual lattice-based cryptography implementations, the polynomial modulus of $\mathbb{Z}[X]$ is a power-of-two cyclotomic polynomial. In such a case, we have a strong connection between lattices, computational number theory and algebraic number theory, since:

$$\mathbb{Z}^n \cong \mathbb{Z}[X]/\langle \Phi_{2n}(x) \rangle \cong \mathcal{O}_K \subset K = \mathbb{Q}[\xi_{2n}] \tag{2.26}$$

where $\Phi_n(x)$ is the n-cyclotomic polynomial and ξ_n is the n-th root of unity – with n being a power of two. In particular, there is a beautiful connection between the "geometry" of roots of unity (Figure 54) and the generated lattice that is (a rotation of) $\mathbb{Z}^{n/2}$.

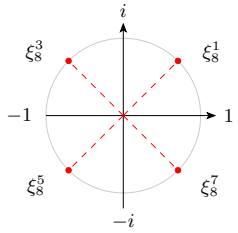


Figure 54: The 4 roots of unity for $X^4 + 1$

Consider the field extension using the fourth root of unity $\mathbb{Q}[\xi_4]$, where $\xi_4 = i$ is the root of $\Phi_4(X) = X^2 + 1$. The Minkowski embedding sends $\xi_4 = i$ to the two possible roots of $X^2 + 1$, namely

$$\sigma_1 : i \to i$$
$$\sigma_2 : i \to -i$$

so we obtain a lattice whose basis is $B = (\boldsymbol{b}_1, \boldsymbol{b}_2) \in \mathbb{C}^{2\times 2}$, and the two basis vectors

$$\boldsymbol{b}_1 = \sigma(1) = (\sigma_1(1), \sigma_2(1)) = (1, 1) \in \mathbb{C}^2$$
$$\boldsymbol{b}_2 = \sigma(i) = (\sigma_1(i), \sigma_2(i)) = (i, -i) \in \mathbb{C}^2.$$

As a result, we precisely obtain a space isomorphic to $\mathbb{Z}^2$. Just like in Example 5, we now aim to build a sublattice of $\mathbb{Z}^n$, which can be used as a starting point to hide secrets. We will do that by considering $\Phi_8(X) = X^4 + 1$ and also by introducing a way to compute the Minkowski embedding via the so-called Vandermonde matrix in the following example.

Example 8 (*Polynomial → nega-cyclic lattice*)

Consider the ring

$$\mathcal{R} = \mathbb{Z}[X]/\langle X^4 + 1\rangle.$$

Notice that $\mathcal{R} \cong \mathcal{O}_K$ where $K = \mathbb{Q}[\xi_8]$, and $\xi_8 = e^{i\pi/8}$. So at this point we have $\mathbb{Z}^4$, and we want to build a lattice $\mathcal{L} \subset \mathbb{Z}^4$. Fix an arbitrary ideal, e.g.,

$$h(x) = 1 + 4x - 2x^3$$

and consider the four roots of $X^4 + 1$:

$$\alpha_1 = \xi_8,\ \alpha_2 = \xi_8^3,\ \alpha_3 = \xi_8^5,\ \alpha_4 = \xi_8^7.$$

The Minkowski embedding sends $h(X)$ to the following:

$$\sigma(h(X)) = (h(\alpha_1), h(\alpha_2), h(\alpha_3), h(\alpha_4)) \in \mathbb{C}^4,$$

which can be computed as a standard vector matrix multipli-

cation using the following, called the Vandermonde matrix:

$$V = \begin{pmatrix} 1 & \xi_8 & \xi_8^2 & \xi_8^3 \\ 1 & \xi_8^3 & \xi_8^6 & \xi_8^9 \\ 1 & \xi_8^5 & \xi_8^{10} & \xi_8^{15} \\ 1 & \xi_8^7 & \xi_8^{14} & \xi_8^{21} \end{pmatrix} \in \mathbb{C}^{4\times 4},$$

where the i-th row corresponds to evaluating $h(\alpha_i)$ (or evaluating $h(X)$ at root ξ_8^{2i-1}) so to obtain the actual embedding, we could simply compute

$$\sigma(h(X)) = V \cdot (1, 4, 0, -2) \in \mathbb{C}^4$$

Under the coefficients encoding, we could generate a lattice basis simply by computing all the powers of X, for instance $\boldsymbol{b}_1 = (1, 4, 0, -2) \in \mathbb{Z}^4$, then $\boldsymbol{b}_2 = (2, 1, 4, 0) \in \mathbb{Z}^4$ and so on. Since we are working modulo $X^4 = -1$, when an element gets rotated, it will have their sign changed.

Defining the basis with respect to the $(1, X, ..., X^{n-1})$ power basis will require quadratic complexity $\Omega(n^2)$ to perform a multiplication (required, e.g., when generating pk for a public key scheme). On the other hand, by defining the basis with respect to $(1, \xi_{2n}, \ldots, \xi_{2n}^{n-1})$ the same operation is replaced by a single coefficient-wise multiplication – as shown in (2.25) – thus with complexity $\mathcal{O}(n \log(n))$ under the Number theoretic transform (NTT) [Pol71].

In summary, ideal lattices over cyclotomic polynomial represent a natural choice for lattice-based cryptosystems as they are mostly secure (we will go into details in the next section) and are very efficient, with respect to their plain unstructured counterparts.

Security consequences on the usage of algebraic lattices. It has been shown that there is a trade off in terms of security and structure. Some types of attacks [Cra+16; CDW17; PHS19] showed that in lattices whose bases have a strong algebraic structure it is easier to solve certain hard problems. Although there are attempts to use module lattices to construct fully homomorphic encryption schemes [Muk+23], most of them are based on the ring version of LWE, which is defined over ideal lattices. We can define a "spectrum of hardness" as shown in Figure 55.

[Pol71]: Pollard (1971), *'The Fast Fourier Transform in a Finite Field'* . In: Mathematics of Computation

[Cra+16]: Cramer, Ducas, Peikert, and Regev (2016), *'Recovering Short Generators of Principal Ideals in Cyclotomic Rings'* . In: Advances in Cryptology – EUROCRYPT 2016

[CDW17]: Cramer, Ducas, and Wesolowski, *'Short Stickelberger Class Relations and Application to Ideal-SVP'* . In: EUROCRYPT '17

[PHS19]: Pellet-Mary, Hanrot, and Stehlé, *'Approx-SVP in Ideal Lattices with Pre-processing'* . In: EUROCRYPT '19

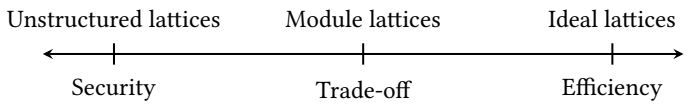


Figure 55: Security spectrum of commonly used family of lattices in cryptography

As an example, we can easily prove that solving $\text{GapSVP}_{\sqrt{n}}$ over ideal lattices is easy. Consider the shortest vector $\boldsymbol{v} \in \mathcal{L}$, and since we are considering an ideal lattice, we can define $\mathcal{L}'$ as the lattice generated by all the nega-cyclic rotations of $\boldsymbol{v}$ (i.e., $\boldsymbol{v}$, $\boldsymbol{v} \cdot X$, $\boldsymbol{v} \cdot X^2$ and so on). Clearly, $\mathcal{L}' \subseteq \mathcal{L}$, so we have that:

$$\det(\mathcal{L}) \leq \det(\mathcal{L}') \leq \|\boldsymbol{v}\|^n = \lambda_1(\mathcal{L})^n.$$

Remark that by Minkowski theorem, $\lambda_1 \leq \sqrt{n}(\det(\mathcal{L}))^{1/n}$, so by putting these two pieces together we obtain that:

$$\det(\mathcal{L})^{1/n} \leq \lambda_1 \leq \sqrt{n}(\det(\mathcal{L}))^{1/n}.$$

Therefore we can simply solve $\text{GapSVP}_{\sqrt{n}}$ by computing the determinant of the lattice[27].

27: Nevertheless, SVP_γ remains hard, as far as we know

2.4.2 Ring-SIS

One simple application of structured lattices concerns the SIS problem. The one-way function induced by this problem is indeed defined as: given a random $A \in \mathbb{Z}_q^{m \times n}$, with $m > n \log(q)$, sample a uniform $\boldsymbol{z} \sim \{0,1\}^n$ and output $\boldsymbol{y} = A^t \boldsymbol{z} \bmod q$. The challenge is to recover the secret $\boldsymbol{z}$.

To represent A, we require to represent $m \times n$ numbers, and typically the order of magnitude for such values requires to store tens of thousands of values. Moreover, the $A^t \boldsymbol{z}$ multiplication can be expensive for large dimensions: this is not suitable for real-world scenarios. One approach could be to use nega-cyclic matrices to build the actual A. In particular, each square submatrix of A could be defined by a polynomial modulo a cyclotomic polynomial so that the whole A matrix can be defined using only m values!

Definition 46 (*Ring-SIS*)

Given ℓ random polynomials $a_1, a_2, \ldots, a_\ell \in \mathbb{Z}_q[X]/(X^n + 1)$, find polynomials $z_1, z_2, \ldots, z_\ell$ with small entries such that:

$$a_1 z_1 + a_2 z_2 + \ldots a_\ell z_\ell \equiv 0 \bmod X^n + 1.$$

We give a brief example in the following.

Example 9 (*Efficiency of Ring-SIS*)

Consider the following toy example, where A^t is defined as

$$A^t = \begin{pmatrix} 2 & 1 & -3 & 7 & 0 & 2 & -1 & 5 \\ -7 & 2 & 1 & -3 & -5 & 0 & 2 & -1 \\ 3 & -7 & 2 & 1 & -1 & -5 & 0 & 2 \\ -1 & 3 & -7 & 2 & -2 & -1 & -5 & 0 \end{pmatrix}$$

Observe that it contains two square nega-cyclic submatrices. Define the binary vector $\boldsymbol{z}$ is

$$\boldsymbol{z} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

We could rewrite the $A^t\boldsymbol{z}$ multiplication using polynomials as

$$(2 + x - 3x^2 + 7x^3) \cdot (x + x^2) + \\ (2x - x^2 + 5x^3) \cdot (1 + x^3) \bmod (X^4 + 1)$$

We have strong results about the hardness of Ring-SIS, discovered independently by Peikert and Rosen [PR06] and by Lyubashevsky and Micciancio [LM06].

Theorem 7 (*Ideal SVP$_\gamma \leq$ Ring-SIS* [PR06; LM06])

For any power of two n, integer $\ell \geq 1$, and integer modulus $q \geq 2n^2\ell$, SVP$_\gamma$ over the ideal lattice in the ring $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ can be efficiently reduced to Ring-SIS over $\mathcal{R}$, where $\gamma = \ell \cdot poly(n)$.

As a natural consequence, it is interesting to study the hardness of SVP$_\gamma$ over $\mathcal{R}$. There are already cited works [Cra+16; CDW17; PHS19] that allow one to solve approximate SVP over ideal lattices with a large approximation factor (roughly $\gamma = 2^{O(\sqrt{n})}$). Nevertheless, it is not clear whether these attacks can be extended to Ring-SIS, for different reasons: (i) the approximation factor is too large to use the reduction in Theorem 7, and (ii) Ring-SIS is not really a problem over an ideal lattice but its solution consists of a vector of ℓ ring elements – technically speaking, this is a problem over rank ℓ modules. It is therefore still unclear how to use such attacks in the Ring-SIS setting.

To summarize, this compact definition of SIS is, at the current state, as hard as its unstructured version, with the advantage of being computable in linear $\mathcal{O}(n)$ time.

2.4.3 Ring-LWE

Similarly to what has been proposed for Ring-SIS, it is possible to define also a ring version of LWE which is called Ring-Learning with errors (RLWE). It has been firstly introduced in 2010 by Lyubashevsky, Peikert and Regev [LPR10]. We define the ring $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ with n being a power of two[28] and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Therefore, elements of $\mathcal{R}_q$ are polynomials of degree at most $n - 1$ and coefficients in $\mathbb{Z}_q$. As a plus, operations in this ring can be performed very fast using FFT-like algorithms (e.g., multiplication can be done in $O(n \log(n))$).

[LPR10]: Lyubashevsky, Peikert, and Regev, *'On Ideal Lattices and Learning with Errors over Rings'* . In: EUROCRYPT '10

28: It is a nice coincidence that cyclotomic polynomials with $n = 2^k$ are in the "nice" shape $X^n + 1$ that makes proofs possible and also allow to implement things efficiently with FFT-like procedures due to n being a power of two

The definition of RLWE follows the one from LWE (see Equation 2.20), but it replaces the "unstructured" $\mathbb{Z}$ with $\mathcal{R}$ – notice that we use the "vector" notation with polynomials to indicate that they contain multiple elements:

$$\begin{aligned} \boldsymbol{a}_1 &:= \mathcal{R}_q \ , \ \boldsymbol{b}_1 \equiv \boldsymbol{a}_1 \cdot \boldsymbol{x} + s_1 \in \mathcal{R}_q \\ \boldsymbol{a}_2 &:= \mathcal{R}_q \ , \ \boldsymbol{b}_2 \equiv \boldsymbol{a}_2 \cdot \boldsymbol{x} + s_2 \in \mathcal{R}_q \\ &\vdots \\ \boldsymbol{a}_m &:= \mathcal{R}_q \ , \ \boldsymbol{b}_m \equiv \boldsymbol{a}_m \cdot \boldsymbol{x} + s_m \in \mathcal{R}_q \end{aligned} \tag{2.27}$$

Also in this case we can define two flavors of RLWE:

- (Search-RLWE) Given the set of all $(\boldsymbol{a}_i, b_i)$, recover either $\boldsymbol{x} \in \mathcal{R}_q$ or $\boldsymbol{s} \in \mathcal{R}_q$.
- (Decision-RLWE) Distinguish between a uniformly random pair (a_i, b_i) and a RLWE sample $(\boldsymbol{a}_i, b_i) \in \mathcal{R}_q \times \mathcal{R}_q$

Notice that, in the case of LWE, one generates a single sample b_i, starting from a vector $\boldsymbol{a}_i$, a secret $\boldsymbol{x}$ and an error e_i. Under this setting, the structure of $\mathbb{Z}[X]/(X^n + 1)$ allows one to generate n elements for each multiplication, since the inner product $\langle \cdot, \cdot \rangle$ has been replaced by a polynomial multiplication.

Remark 4 (*Compactness*)

By picking n LWE samples $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots \boldsymbol{a}_n$, and by multiplying them all by $\boldsymbol{x}$, we perform a "noisy" linear transformation (which costs one matrix-vector multiplication). By picking a single RLWE sample, we perform a linear transformation at the cost of a polynomial multiplication. Notice that a RLWE sample is a polynomial in $\mathbb{Z}[X]/(X^n + 1)$, therefore in matrix terms a nega-cyclic matrix. Therefore a single RLWE sample stands for n (specific) LWE samples.

Since it will be heavily used in Chapter 3 for standard FHE constructions, we give a formal definition of Decision-RLWE.

Definition 47 (*Decision-RLWE*)

For a security parameter λ, let $p(X)$ be a power of two cyclotomic polynomial Φ_m with degree $deg(p) = \phi(m)$ depending on λ, and set $\mathcal{R} = \mathbb{Z}[X]/p(X)$. Let $q = q(\lambda) \geq 2$ be an integer. For a random element $\boldsymbol{s} \in \mathcal{R}_q$ and a distribution $\chi = \chi(\lambda)$ over $\mathcal{R}$, the Decision-RLWE problem asks to distinguish between a distribution obtained by choosing a uniformly random element $\boldsymbol{a} \sim \mathcal{R}_q$ and a noise term $\boldsymbol{e} := \chi$ outputting $(\boldsymbol{a}, [\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_q)$ and the uniform distribution over $\mathcal{R}_q^2$.

We observe that, for efficiency reasons, most schemes use power of two cyclotomic polynomials. It has been shown [LPR10] that there exists a (quantum) reduction from worst case approximate SVP over ideal lattices to RLWE, similarly to the reduction from Ring-SIS. In particular, the original reduction involved both search-RLWE and decision-RLWE[29], while some years later a new reduction [PRS17] skipped entirely search-LWE.

29: We will not go into the details, but reducing search-to-decision for RLWE is not as simple as in the LWE case, as changing one element of the secret – an operation performed in that reduction – has an impact in all the coefficients of the output ring element

Theorem 8 (*Hardness of RLWE* [LPR10])

For any power of two n, ring $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$, an integer $q \equiv 1 \bmod n$, and $\beta > \sqrt{n \log(n)}$, there is an efficiently samplable distribution χ that outputs elements of $\mathcal{R}$ of length at most β with overwhelming probability, such that if there exists an efficient algorithm that solves $\mathrm{RLWE}_{n,q,\chi}$, then there is an efficient quantum algorithm for solving $n \cdot (q/\beta)$-approximate worst case of SVP for ideal lattices over $\mathcal{R}$.

[PRS17]: Peikert, Regev, and Stephens-Davidowitz, *'Pseudorandomness of ring-LWE for any ring and modulus'* . In: STOC 2017

We refer to a "unified" view of different algebraic LWE definitions by Peikert and Pepin [PP19].

[PP19]: Peikert and Pepin, *'Algebraically Structured LWE, Revisited'* . In: TCC '19

2.5 Lattice isomorphism problem

One interesting problem, which will be used in Chapter 4, concerns finding transformations between isomorphic lattices. Two lattices are said to be isomorphic if they differ by an isometry, namely a transformation that preserves the distances [HR14]. In practice, an isometry can be a rotation or a scaling. Let us first introduce the definition of the problem, called *Lattice isomorphism problem* (LIP) which was recently introduced as a cryptographic building block concurrently by Ducas and

[HR14]: Haviv and Regev, *'On the Lattice Isomorphism Problem'* . In: SODA '14

[DvW22]: Ducas and van Woerden, *'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'* . In: EUROCRYPT '22

van Woerden [DvW22] and Bennett, Ganju, Peetathawatchai and Stephens-Davidowitz [Ben+23b]. We report for the sake of completeness some concepts that will be useful to better understand later reasonings following the notation and constructions from [DvW22], although we refer to the original paper for detailed arguments.

Definition 48 (LIP**, *lattice version*)**

Given two isomorphic lattices $\mathcal{L}, \mathcal{L}' \subset \mathbb{R}^n$, find an orthonormal transformation $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}' = O \cdot \mathcal{L}$.

We saw that two lattices are equal if their bases differ by a unimodular transformation, namely $\mathcal{L}(B) = \mathcal{L}(B \cdot U)$ for some unimodular matrix $U \in \mathrm{GL}_n(\mathbb{Z})$. Therefore, we can restate the problem in terms of the bases. LIP asks, given B and $B' = OBU$, for $O \in \mathcal{O}_n(\mathbb{R})$ and $U \in \mathrm{GL}_n(\mathbb{Z})$, to recover either O or U. The presence of both transformations is what makes the problem hard, even by considering the trivial $\mathbb{Z}^n$ lattice. A LIP instance is constructed as follows: consider some lattice $\mathcal{L}$ generated by $B \in \mathbb{R}^{n \times n}$. For instance $B = I_n$ is the identity (Figure 56).

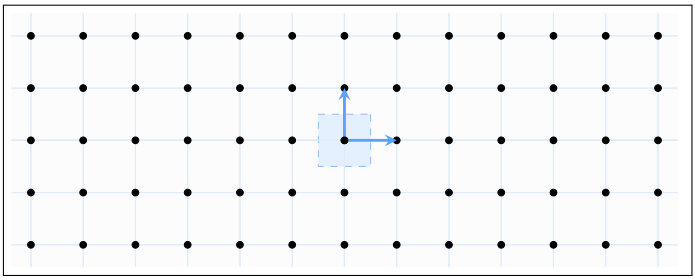


Figure 56: The simplest lattice $\mathbb{Z}^n$, with $n = 2$, generated by $B = I_n$

Now, apply some sufficiently bad unimodular transformation (we will later see how to sample one), to obtain $\mathcal{L}(B \cdot U)$ (Figure 57).

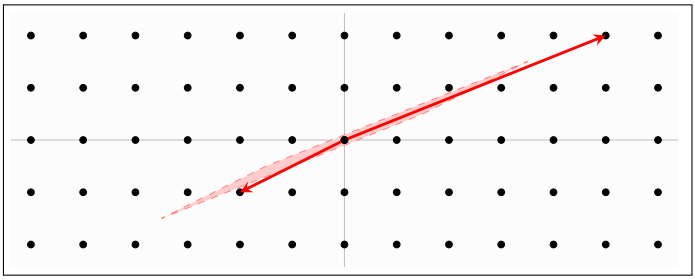


Figure 57: The simplest lattice $\mathbb{Z}^n$, with $n = 2$, generated by $B \cdot U$ for $B = I_n$

To conclude, sample a random orthonormal transformation

O and apply it to the lattice to obtain the new basis OBU, as illustrated in Figure 58.

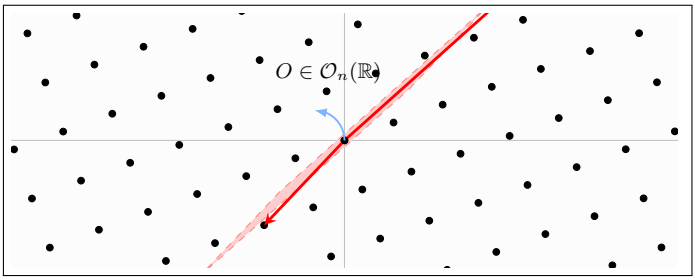


Figure 58: The rotated simplest lattice $\mathbb{Z}^n$, with $n = 2$, generated by $O \cdot B \cdot U$ for $B = I_n$

Notice that, if the lattice $\mathcal{L}(B)$ has a short vector $\boldsymbol{s} \in \mathcal{L}(B)$, then the rotated lattice will also have the same vector, but rotated as $O\boldsymbol{s} \in O\mathcal{L}(B)$. This gives a very natural and elegant way to hide the shortest vector of *any* lattice.

However, there is a problem related to the fact that, by using these objects in cryptography, we need to store very large values as the orthonormal transformation has real entries.

Quadratic forms. A *quadratic form* is a positive definite real symmetric matrix. We will not discuss about any detailed argument regarding quadratic forms, nonetheless consider the Gram map γ:

$$\gamma : \mathrm{GL}_n(\mathbb{R}) \mapsto \mathcal{S}_n^{>0}(\mathbb{R}),$$

sending some lattice basis in $\mathrm{GL}_n(\mathbb{R})$ to a Gram matrix in $\mathcal{S}_n^{>0}(\mathbb{R})$, namely the set of real symmetric positive definite matrices of size $n \times n$. This map is realized by sending a lattice basis B to the quadratic form $Q = B^t B$. Now, observe that the preimages of $\gamma(B)$ are all the possible $O \cdot B$ for $O \in \mathcal{O}_n(\mathbb{R})$, and observe indeed that:

$$(O \cdot B)^t (O \cdot B) = B^t \cdot O^t \cdot O \cdot B = B^t B,$$

since by definition $O^t O = I_n$. By declining this observation under LIP terms, we have that for $Q = B^t B$ and $B' = OBU$:

$$\begin{aligned} Q' &= (B')^t B' = (OBU)^t (OBU) = U^t B^t O^t OBU \\ &= U^t B^t BU = U^t Q U. \end{aligned}$$

We say that Q, Q' are equivalent if such a unimodular $U \in \mathrm{GL}_n(\mathbb{Z})$ exists, and we refer to $[Q]$ as the equivalence class (i.e., the set of all quadratic forms equivalent to Q). This allows us

to restate LIP in a more convenient – but equivalent – way, as we now deal with integer values only. We define two versions of the problem, search and distinguish respectively.

Definition 49 (sLIP^{Q}, *quadratic forms version*)

For a quadratic form $Q \in \mathcal{S}_n^{>0}(\mathbb{R})$, the wc-sLIP^{Q} *problem is, given any quadratic form Q' equivalent to Q, to find a unimodular $U \in GL_n(\mathbb{Z})$ such that $Q' = U^t Q U$.*

Definition 50 (ΔLIP^{Q_0,Q_1}, *quadratic forms version*)

For two quadratic forms $Q_0, Q_1 \in \mathcal{S}_n^{>0}(\mathbb{R})$, the wc-$\Delta$LIP^{Q_0,Q_1} *is, given any quadratic form Q' equivalent to Q_b, where $b \in \{0,1\}$ is a uniform random bit, to recover b.*

These definitions concern the problems in their worst case, although for useful cryptographic applications we usually define average cases.

An average case of LIP. We introduce a distribution that is used to define the average case of the different LIP problems.

Definition 51 (*Gaussian form distribution $\mathcal{D}_s([Q])$*)

For any quadratic form $Q \in \mathcal{S}_n^{>0}(\mathbb{R})$, we define the Gaussian form distribution *$\mathcal{D}_s([Q])$ over Q, with parameter $s > 0$, algorithmically as follows:*

1. *Let $C := 1 - (1 + e^{-\pi})^{-1} > 0$ and $m := \lceil \frac{2n}{C} \rceil$. Sample m vectors $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m)$ from $\mathcal{D}_{Q,s}$ and repeat until the rank of Y has full rank n.*
2. *$R, U :=$ Extract(Q, Y)*
3. *Return R*

The Extract function takes as input some random Y (according to a discrete Gaussian distribution) and returns a unimodular transformation using the Hermite normal form. We show that such a function exists in the following lemma.

Lemma 21 ([DvW22, Lemma 3.1])

There is a polynomial time algorithm $(R, U) :=$ Extract(Q, Y) that on input a quadratic form Q and vectors $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \in \mathbb{Z}^{m \times n}$ such that rank$(Y) = m$, outputs a transformation $U \in GL_n(\mathbb{Z})$ and a quadratic form $R = U^t Q U$ such that $\|\tilde{B_R}\| \leq \max_i \|\boldsymbol{y}_i\|_Q$.

Notice that B_R refers to the (Cholesky) decomposition of R as $R = B_R^t B_R$. We omit details and refer to the proof that can be found in [DvW22]. We therefore conclude by presenting the search and distinguish LIP problems, but stated with respect to the $\mathcal{D}_s([Q])$ distribution .

Definition 52 (ac-sLIP$_s^Q$**)**

For a quadratic form $Q \in \mathcal{S}_n^{>0}(\mathbb{R})$ and $s > 0$, the ac-sLIP$_s^Q$ *problem is, given a quadratic form Q' sampled as $Q' \sim \mathcal{D}_s([Q])$, to find a unimodular $U \in GL_n(\mathbb{Z})$ such that $Q' = U^t Q U$.*

Definition 53 (ac-ΔLIP$_s^{Q_0,Q_1}$**)**

For two quadratic forms $Q_0, Q_1 \in \mathcal{S}_n^{>0}(\mathbb{R})$ and $s > 0$, the ac-ΔLIP$_s^{Q_0,Q_1}$ *problem is, given a quadratic form sampled as $Q' \sim \mathcal{D}_s([Q_b])$, where $b \in \{0,1\}$ is a uniform random bit, to recover b.*

Constructions

Modern RLWE-based schemes

3

To start the "constructions" part, we will begin by presenting the modern RLWE-based constructions of FHE schemes. We will do that by using material from the previous chapter.

We begin by formally defining the syntax of a FHE scheme. In particular, we start by considering BGV [BGV12], which is a classic FHE scheme where the message space is $\mathcal{M} = \mathbb{Z}_t$, if not specified we refer to the binary plaintext space $\mathbb{Z}_2$. Additionally, the model of computation is given by arithmetic modulo t. By having access to addition and multiplication over $\mathbb{Z}_t$ we have an extremely general computational model. Not as strong as *any* programming language, since we can not change the flux of execution according to the input values, since they are encrypted, so we do not have access to, e.g., the *selection* control structure [BJ66]. We also refer to [Gol+13] for some interesting discussions and results about the computational power of encrypted systems.

[BJ66]: Böhm and Jacopini (1966), *'Flow diagrams, turing machines and languages with only two formation rules'* . In: Commun. ACM

[Gol+13]: Goldwasser, Kalai, Popa, Vaikuntanathan, et al., *'How to Run Turing Machines on Encrypted Data'* . In: CRYPTO '13

An FHE scheme $\mathcal{E}$ consists of a tuple of four efficient (i.e., probabilistic polynomial-time) randomized algorithms:

$$\mathcal{E} := (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval}),$$

defined as follows:

- **Key generation:** $\text{Keygen}(1^\lambda) \to (pk, sk, ek)$. Given the security parameter λ, the key generation algorithm outputs (i) a *public encryption key* pk, which allows any party to generate an encryption of $m \in \mathcal{M}$, (ii) a *secret decryption key* sk, which is kept private by the user, and lastly (iii) an *evaluation key* ek, which is made available to any party performing homomorphic computations on ciphertexts encrypted under sk.
- **Encryption:** $\text{Enc}_{pk}(\mu) \to \boldsymbol{c}$. Given the public key pk and a message bit $\mu \in \mathcal{M}$, the encryption algorithm outputs a ciphertext $\boldsymbol{c}$ encrypting μ.
- **Decryption:** $\text{Dec}_{sk}(\boldsymbol{c}) \to \mu$. Given the secret key sk and a ciphertext $\boldsymbol{c}$, the decryption algorithm recovers the plaintext bit μ.
- **Evaluation:** $\text{Eval}_{ek}(F, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell) \to \tilde{\boldsymbol{c}}$. The evaluation algorithm takes as input (i) the evaluation key, ek, (ii)

some arithmetic circuit $F : \mathbb{Z}_t^\ell \to \mathbb{Z}_t$ and (iii) ℓ ciphertexts $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell$ encrypting the inputs to F. It outputs a ciphertext $\tilde{\boldsymbol{c}}$ that encrypts the circuit output, i.e.,

$$\tilde{\boldsymbol{c}} = \mathrm{Eval}_{ek}(F, (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_\ell)).$$

We will not specify here to which class the circuit F belongs, as it depends on the scheme and the parameters. Theoretically, with bootstrapping [Gen09b], we can take *any* arithmetic circuit (i.e., composed of additions and multiplications) F. Otherwise, only some fixed-depth one, depending on the parameters of the scheme.

We introduce a few definitions that will be useful to simplify later arguments. We also observe that all the considered schemes share the same key generation process (from LWE). From a certain perspective, the decryption circuit is also the same (there might be minor changes depending on how data is encoded). In particular, we will deal with FHE schemes that are *compact* and *semantically secure*.

Definition 54 (*Compactness, informal*)

An FHE scheme $\mathcal{E} = (\mathrm{Keygen}, \mathrm{Enc}, \mathrm{Dec}, \mathrm{Eval})$ is called compact *if, for any security parameter λ, any keypair $(pk, sk) := Keygen(1^\lambda)$, any circuit C of polynomial size in λ, and any ciphertexts $\boldsymbol{c}_i := Enc_{pk}(\boldsymbol{m}_i)$,*

$$\boldsymbol{c}_{out} := Eval_{ek}(C, (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell)) \quad \Rightarrow \quad |\boldsymbol{c}_{out}| = |\boldsymbol{c}_i|$$

Therefore, the size of the resulting ciphertext does not change regardless of the choice of circuit C, its size, depth, or the number of input ciphertexts ℓ.

For security, we will assume our FHE schemes are simply public-key encryption schemes and use the following security definition.

Definition 55 (*Semantic security (IND-CPA)*)

Let $\Pi = (Keygen, Enc, Dec)$ be a public-key encryption scheme. We say that Π is semantically secure *(or* IND-CPA secure*) if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, there exists a negligible function $negl(\cdot)$ such that*

$$\left| \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(1^\lambda)$ is defined as follows:

1. *The key generation algorithm is run to obtain $(pk, sk) \leftarrow$*

Keygen(1^λ).
2. *The adversary* $\mathcal{A}$ *is given* pk *and outputs two messages of equal length* (m_0, m_1).
3. *A random bit* $b \sim \{0,1\}$ *is chosen, and the ciphertext is computed as* $c^* \leftarrow Enc(pk, m_b)$.
4. *The adversary* $\mathcal{A}$ *is given* c^* *and outputs a bit* $b' \in \{0,1\}$.
5. *The experiment outputs* 1 *if* $b' = b$, *and* 0 *otherwise.*

Intuitively, no efficient adversary can distinguish encryptions of two chosen messages with non negligible probability better than random guessing.

We will not give formal security proofs for the schemes in this chapter, which can be found in the corresponding papers. However, we will prove the security of the new schemes that we will propose in Chapter 4.

Notice that we will not consider advanced definitions of *security*, introduced specifically for FHE schemes. In particular, Li and Micciancio [LM21] noticed that standard IND-CPA security is not sufficient for approximate FHE schemes (this observation was also extended to exact schemes [Che+24b; Che+24a]), and presented a strong attack that leads to full key recovery in a short time. They proposed a new definition of security, called IND-CPA$^{\text{D}}$. Their attack is based on a simple observation that we present informally and briefly. An attacker can ask for an encryption of 0, that is a pair $(a, b = a \cdot s + e \bmod q)$ (in whatever representation, LWE or RLWE), and then asks a decryption of the obtained ciphertext (which will be very close to zero, but not really zero) and they can compute

$$b - \text{Dec}(\boldsymbol{c}) = a \cdot s + e - m + e = a \cdot s \bmod q$$

that provides a linear equation in the secret key. By collecting n equations – and ignoring complexities coming from e.g., encoding/decoding processes – of this form we can recover the secret by Gaussian elimination with high probability. We will not discuss this class of attacks further, but we refer the interested reader to the previously referenced papers. For simplicity, in the rest of the thesis, we can assume that the results of the computations will never be shared by the client.

We now introduce some "implementative" concepts that will be required to better understand how the implementation of such schemes work in practice.

[LM21]: Li and Micciancio, *'On the Security of Homomorphic Encryption on Approximate Numbers'* . In: EUROCRYPT '21

[Che+24b]: Cheon, Choe, Passelègue, Stehlé, et al., *'Attacks Against the IND-CPA^D Security of Exact FHE Schemes'* . In: CCS '24

[Che+24a]: Checri, Sirdey, Boudguiga, and Bultel, *'On the Practical IND-CPA^D Security of "exact" and Threshold FHE Schemes and Libraries'* . In: CRYPTO '24

3.1 Efficient representations of polynomials

As it will be shown later, most leveled schemes (e.g., BGV, BFV, CKKS) deal with *very large* moduli. As a consequence, dealing with such gigantic values makes it hard to implement the procedures efficiently on standard CPUs. One clever way to deal with this problem is to use the Double Chinese remainder theorem (DCRT) representation. This (perhaps misleading) term refers to the application of two CRT related concepts: the Residue number system (RNS) and the Number theoretic transform (NTT). We remark that both can be considered as "optimizations" to speed up computations and not as required procedures for security or the correctness of the schemes.

Residue number system. The RNS takes advantage of the ring isomorphism

$$\mathbb{Z}_q \cong \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_k}, \quad \text{where } q = \prod_{i=1}^{k} p_i \tag{3.1}$$

and represents an integer $x \in \mathbb{Z}_q$ with respect to a set of pairwise coprime moduli $p_1, p_2, \ldots, p_k$ as the tuple of residues

$$(x \bmod p_1, x \bmod p_2, \ldots, x \bmod p_k),$$

Under the RNS representation, addition and multiplication of integers modulo q can be performed independently in each modulus channel, enabling efficient, parallel, and carry-free arithmetic. This property is heavily leveraged in most lattice cryptography to accelerate polynomial and modular operations central to RLWE-based schemes.

In the presented schemes we deal with polynomials in $\mathcal{R}_q = \mathbb{Z}[X]/(X^n + 1)$, and in actual implementations these are represented using smaller polynomials in $\mathcal{R}_{p_i}$ by exploiting that:

$$\mathcal{R}_q \cong \mathcal{R}_{p_1} \times \mathcal{R}_{p_2} \times \cdots \times \mathcal{R}_{p_k}.$$

Example 10 (*Residue number system*)

Define $\mathcal{R} = \mathbb{Z}[X]/(X^4 + 1)$ and modulus $Q = 105 = 3 \cdot 5 \cdot 7$, where all factors $q_1 = 3, q_2 = 5, q_3 = 7$ are pairwise coprime. A polynomial in $\mathcal{R}_Q$ can be decomposed as

$$\mathcal{R}_Q \cong \mathcal{R}_{q_1} \times \mathcal{R}_{q_2} \times \mathcal{R}_{q_3}$$

Consider the following $a(X)$ polynomial:

$$a(X) = 23 + 47X + 88X^2 + 101X^3 \in \mathcal{R}_Q,$$

we can compute all the residues:

Coefficient	$\mathbb{Z}_{105}$	mod 3	mod 5	mod 7
X^0	23	2	3	2
X^1	47	2	2	5
X^2	88	1	3	4
X^3	101	2	1	3

Table 5: Coefficient-wise CRT decomposition of $a(X)$

So the image under CRT of $a(X)$ is the triple of polynomials $(a_1(X), a_2(X), a_3(X))$, where:

$$a_1(X) = 2 + 2X + X^2 + 2X^3 \in \mathcal{R}_3$$
$$a_2(X) = 3 + 2X + 3X^2 + X^3 \in \mathcal{R}_5$$
$$a_3(X) = 2 + 5X + 4X^2 + 3X^3 \in \mathcal{R}_7$$

Number theoretic transform. The RNS tackles the problem of dealing with large integer numbers, while the role of the Number theoretic transform is to deal with polynomial multiplications. In particular, the NTT is a finite field version of the Fourier transform, and we define it as follows.

Definition 56 (*NTT transformation*)

Let $n > 1$ and q be a prime with $q \equiv 1 \bmod 2n$, and let ω be a primitive n-th root of unity in $\mathbb{Z}_q$. For a vector $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$, $\mathrm{NTT}(\boldsymbol{a})$ is the vector $\hat{\boldsymbol{a}} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1})$ where:

$$\hat{a}_i = \sum_{j=0}^{n-1} \omega^{ij} a_j \bmod q.$$

The main idea is that, in general, performing a naïve multiplication between two polynomials has $\mathcal{O}(n^2)$ complexity, while it has a linear complexity (it is a point-wise multiplication) for two NTT transformed polynomials, so the cost of the multiplication simply becomes the cost of the NTT transformation, plus its inverse INTT.

Notice that this transformation is similar to the Minkowski embedding (Definition 44), which was implemented using a

Vandermonde matrix as

$$V = \begin{pmatrix} \xi_{2n}^0 & \xi_{2n}^1 & \cdots & \xi_{2n}^{n-1} \\ \xi_{2n}^0 & \xi_{2n}^3 & \cdots & \xi_{2n}^{3(n-1)} \\ \xi_{2n}^0 & \xi_{2n}^5 & \cdots & \xi_{2n}^{5(n-1)} \\ \xi_{2n}^0 & \xi_{2n}^7 & \cdots & \xi_{2n}^{7(n-1)} \end{pmatrix} \in \mathbb{C}^{n \times n},$$

where each row refers to the evaluation of the polynomial in one root – remark that the roots of $X^n + 1$, with n being a power of two, are the n powers of the primitive $\xi_{2n} = e^{2i\pi/n}$. Since we are working in $\mathcal{R}_q$, this transformation must be defined using an integer matrix by considering the primitive root of unity as the value ω such that $\omega^n = 1 \bmod q$.

Notice that in such case $\omega \in \mathbb{Z}_q$, making the implementation easier as it is based on integer computations only. By building the corresponding Vandermonde matrix with respect to ω, we can easily define the NTT transformation as

$$\text{NTT}(a(X)) = V \cdot \boldsymbol{a},$$

with $\boldsymbol{a} \in \mathbb{Z}_q^n$ being the vector of coefficients of $a(X)$.

Example 11 (*NTT transformation*)

Consider the case where $\mathcal{R} = \mathbb{Z}[X]/(X^4 + 1)$ and computations are performed modulo 17, i.e., in $\mathcal{R}_{17}$. We need to find ω such that

$$\omega^n + 1 = 0 \bmod 17.$$

Now observe that $\omega = 9$ is a primitive root of unity modulo 17 (i.e., $9^4 + 1 = 0 \bmod 17$). We can therefore define the Vandermonde matrix as follows:

$$V = \begin{bmatrix} 9^0 & 9^1 & 9^2 & 9^3 \\ 16^0 & 16^1 & 16^2 & 16^3 \\ 8^0 & 8^1 & 8^2 & 8^3 \\ 1^0 & 1^1 & 1^2 & 1^3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 9 & 13 & 15 \\ 1 & 16 & 1 & 16 \\ 1 & 8 & 13 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \in \mathbb{Z}_{17}^{4 \times 4}$$

Therefore a polynomial $a(X) \in \mathcal{R}_{17}$ can be transformed in NTT form by simply evaluating $V \cdot \boldsymbol{a} \in \mathcal{R}_{17}$.

After the application of NTT, we can evaluate the point-wise multiplication between two polynomials, and invert the result back to the "standard" form to see the result (like shown in

Example 7). In order to invert the NTT transformation, we can simply evaluate the inverse of the V matrix, namely:

$$\text{INTT}(a(X)) = V^{-1} \cdot \boldsymbol{a},$$

and notice that

$$\text{INTT}(\text{NTT}(a(X))) = a(X).$$

The inverse Vandermonde matrix V^{-1} is computed like the previous V, but based on the inverse root ω^{-1} (namely x such that $\omega \cdot x = 1 \bmod q$).

Definition 57 (*INTT transformation*)

Let q be a prime with $q \equiv 1 \bmod 2n$, and let ω be a primitive n-th root of unity in $\mathbb{Z}_q$. For a transformed vector $\hat{\boldsymbol{a}} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1}) \in \mathbb{Z}_q^n$, $\text{INTT}(\hat{\boldsymbol{a}})$ is the vector $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ where:

$$a_i = n^{-1} \sum_{j=0}^{n-1} \omega^{-ij} \hat{a}_j \bmod q,$$

and n^{-1} is the modular inverse of n modulo q.

To summarize, in actual implementations usually elements from $\mathcal{R}_q$ are (i) decomposed using RNS and (ii) each small polynomial is transformed using the NTT transformation. In literature, this representation is referred to as DCRT (see Section A.3 of [GHS12b]). We refer the interested reader to [Sei18] for concrete and technical implementation of polynomial multiplication under NTT and to [Sat+23] for a broader introduction to NTT. We also refer the interested reader to the nice and broad introduction to FHE schemes by Ko [Ko25].

3.2 Brakerski-Gentry-Vaikuntanathan (BGV)

The Brakerski-Gentry-Vaikuntanathan (BGV) scheme [BGV12] has opened up the road for the so-called "leveled" type of FHE. It was initially presented as a "bootstrapping-as-an-optimization" alternative of fully homomorphic encryption, and it was one of the first practical schemes based on RLWE. The footprint follows from Regev scheme (refer to Section 2.3.1), but BGV includes many tricks that allow to speed up and make homomorphic operations faster. We define the RLWE-based version for convenience, although it is possible

[GHS12b]: Gentry, Halevi, and Smart, *'Homomorphic Evaluation of the AES Circuit'* . In: CRYPTO '12

[Sei18]: Seiler (2018), Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography (Cryptology ePrint Archive, Paper 2018/039)

[Sat+23]: Satriawan, Syafalni, Mareta, Anshori, et al. (2023), *'Conceptual Review on Number Theoretic Transform and Comprehensive Review on Its Implementations'* . In: IEEE Access

[Ko25]: Ko (2025), The Beginner's Textbook for Fully Homomorphic Encryption

[BGV12]: Brakerski, Gentry, and Vaikuntanathan, *'(Leveled) fully homomorphic encryption without bootstrapping'* . In: ITCS '12

to find a general definition (that also suits plain LWE) in the original paper [BGV12]. Notice that, from now on, when using RLWE, we will use the $\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}$ notation, as it is more coherent to the literature and we are not interested, in this chapter, in giving a lattice perspective to RLWE. We will adapt to the notation used in a modern implementation of BGV by Kim, Polyakov and Zucca [KPZ21].

[KPZ21]: A. Kim, Polyakov, and Zucca, *'Revisiting Homomorphic Encryption Schemes for Finite Fields'* . In: ASIACRYPT '21

Basic definition. Take as input a security parameter λ, define the ring $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ with $n = n(\lambda)$ being a power of two and a number of levels L – namely multiplications – to be supported. Theoretically, we could generate a set of parameters (i.e., keys) for each level $\ell \in (0, L]$, but we allow for circular security and we generate a unique set of parameter for any level L.

We fix modulus $q := q(\lambda) = \prod_{i=1}^{L} q_i$ such that $q_L \mid t$ and – only for the modulus – we use the notation q_i to refer to the product of the first i moduli; thus q_L is the largest modulus and q_1 the smallest one. We also define two Gaussian-like distribution $\chi_{\text{key}}, \chi_{\text{err}}$ over $\mathcal{R}$. Notice that these distributions are set to be as the smallest possible such that they satisfy the security requirements (i.e., such that $n = \Omega(\lambda \cdot \log(q/\beta))$). Differently from public-key schemes, in FHE schemes we always want to inject noise that is as small as possible. The plaintext space is $\mathcal{M} := \mathbb{Z}_t$ for some $t \ll q_1$ and the ciphertext space is $\mathcal{C} := \mathcal{R}_{q_i}^2$, for some $i < L$.

We will use the bold notation $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \in \mathcal{R}_q^2$ for ciphertexts even though they are technically a pair of polynomials. Nevertheless, we consider them as degree 1 polynomials in the form $\boldsymbol{c}_1 + \boldsymbol{c}_2 X$, where each $\boldsymbol{c}_i \in \mathcal{R}_q$ is a polynomial referred to as a vector containing its coefficients.

> **Setup**(λ)
>
> Given a plaintext modulus t and a moduli chain $q = q_1 \mid q_2 \mid \cdots \mid q_L$ such that t and q_L are coprime, with $L > 0$ being the number of evaluable levels, define $n = n(\lambda, q)$ and $\chi_{\text{key}} = \chi_{\text{key}}(\lambda, q), \chi_{\text{err}} = \chi_{\text{err}}(\lambda, q)$ to achieve 2^λ security against RLWE attacks.

The key generation immediately follows from RLWE: the public key is a random ideal q-ary lattice with a (scaled) short vector $\boldsymbol{e}$, the secret key is the information that allows to recover it.

Keygen(n, q_L)

For the secret key, simply sample $\boldsymbol{s} \sim (\chi_{\text{key}})^n$ and set:

$$sk := \boldsymbol{s} \in \mathcal{R}_{q_L},$$

the public key is formed by planting a RLWE instance to a random ideal q-ary lattice, namely:

$$pk := ([\boldsymbol{a} \cdot \boldsymbol{s} + t\boldsymbol{e}]_{q_L}, -\boldsymbol{a}) \in \mathcal{R}_{q_L}^2.$$

Return (pk, sk).

Notice that by evaluating the public key in $X = \boldsymbol{s}$, we precisely obtain the short vector $[t\boldsymbol{e}]_q$, namely:

$$pk(sk) = [\boldsymbol{a} \cdot \boldsymbol{s} + t\boldsymbol{e} - \boldsymbol{a} \cdot \boldsymbol{s}]_q \equiv [t\boldsymbol{e}]_q \in \mathcal{R}_{q_L}.$$

Enc$_{pk}(\boldsymbol{m}, \ell)$

For a message $\boldsymbol{m} \in \mathcal{R}_t$ and a level $0 < \ell \leq L$:

$$\boldsymbol{c} := ([[\boldsymbol{m}]_t + \boldsymbol{u} \cdot \boldsymbol{pk}_1 + t\boldsymbol{e}_1]_{q_\ell}, [\boldsymbol{u} \cdot \boldsymbol{pk}_2 + t\boldsymbol{e}_2]_{q_\ell}) \in \mathcal{R}_{q_\ell}^2$$

where $\boldsymbol{u} \sim \chi_{\text{key}}$ and $\boldsymbol{e}_1, \boldsymbol{e}_2 \sim \chi_{\text{err}}$. Return $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$.

Notice that $\boldsymbol{pk}_1, \boldsymbol{pk}_2$ refer to the first two components of pk, respectively. To decrypt, we simply evaluate $\boldsymbol{c}$ in $X = \boldsymbol{s}$.

Dec$_{sk}(\boldsymbol{c})$

Given the ciphertext $\boldsymbol{c} = \boldsymbol{c}_1 + \boldsymbol{c}_2 X$ at level i and the secret key $sk = \boldsymbol{s}$, return

$$[[\boldsymbol{c}(\boldsymbol{s})]_{q_i}]_t$$

We can see that decryption is correct, since:

$$\begin{aligned} \boldsymbol{c}(sk) &\equiv [[\boldsymbol{m}]_t + \boldsymbol{u} \cdot \boldsymbol{pk}_1 + t\boldsymbol{e}_1 + \boldsymbol{u} \cdot \boldsymbol{pk}_2 \boldsymbol{s} + t\boldsymbol{e}_2 \cdot \boldsymbol{s}]_q \\ &\equiv [[\boldsymbol{m}]_t + t(\boldsymbol{u} \cdot \boldsymbol{e} + \boldsymbol{e}_1 + \boldsymbol{e}_2 \cdot \boldsymbol{s})]_q \\ &\equiv [\boldsymbol{m}]_t + t \cdot \boldsymbol{v} \in \mathcal{R}_q \end{aligned}$$

and we call $\boldsymbol{v}$ the amount of noise. The message is obtained by simply reducing everything modulo t. In particular, decryption is correct as long at the amount of noise $\boldsymbol{v}$ is such that

$$\|\boldsymbol{v}\|_\infty < \frac{q_1}{2t} \tag{3.2}$$

if this holds, then the message $[\boldsymbol{m}]_t$ does not wrap around any modulo q_i. This happens because the noise gets multiplied by t, and in general we want it to be smaller than $q/2t$.

In the case of Regev cryptosystem (refer to Section 2.3.1), the noise must be smaller than $q/4$, and that is a particular case of this scheme, where we set $q/2t$ (namely $q/4$ in case encrypting binary values) as the maximum amount of noise.

Remark 5

Given the correspondence between LWE and RLWE, the same scheme can be defined in plain form using LWE. The latter gives more intepretability. The plain LWE-based version of BGV will be presented in Section 4.2.2 as it will be useful to better understand the lattice properties required to instantiate the scheme. Moreover, in the same chapter, we will present a generalization called Lattice-BGV (Section 4.5.1).

Homomorphic operations. Homomorphic operations follow (almost) immediately. In particular, addition comes for free as for almost any lattice-based scheme (due to the group structure of lattices).

Add($\boldsymbol{c}, \boldsymbol{c}'$)

Given two ciphertexts $[\boldsymbol{c}]_{q_\ell}, [\boldsymbol{c}']_{q_\ell}$ at the same level ℓ, return $\boldsymbol{c}_{\text{add}} := \boldsymbol{c} + \boldsymbol{c}' \in \mathcal{R}^2_{q_\ell}$.

The correctness of addition can easily be verified:

$$\begin{aligned}(\boldsymbol{c} + \boldsymbol{c}')(\boldsymbol{s}) &\equiv \boldsymbol{c}(\boldsymbol{s}) + \boldsymbol{c}'(\boldsymbol{s}) \\ &\equiv [\boldsymbol{m}]_t + [\boldsymbol{m}']_t + t(\boldsymbol{v} + \boldsymbol{v}') \in \mathcal{R}_q\end{aligned}$$

with the amount of noise that simply gets doubled. It is also possible to multiply by a constant.

Mult($\boldsymbol{c}, k$)

Given a ciphertext $[\boldsymbol{c}]_{q_\ell}$ and a constant $k \in \mathcal{R}_t$, return $\boldsymbol{c}_{\text{k}} := k \cdot \boldsymbol{c} \in \mathcal{R}^2_{q_\ell}$

Correctness follows from addition (in fact, multiplication by a constant can be seen as a particular case of k self-additions). Also multiplications of two ciphertexts are supported.

Mult($\boldsymbol{c}, \boldsymbol{c}'$)

Given two ciphertexts $[\boldsymbol{c}]_{q_\ell}, [\boldsymbol{c}']_{q_\ell}$ at the same level ℓ, return $\boldsymbol{c}_{\text{mul}} := (\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1, \boldsymbol{c}_1 \cdot \boldsymbol{c}'_2 + \boldsymbol{c}_2 \cdot c'_1, \boldsymbol{c}_2 \cdot \boldsymbol{c}'_2) \in \mathcal{R}^3_{q_\ell}$

Notice that multiplication is performed considering a ciphertext $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \in \mathcal{R}^2_{q_\ell}$ as a polynomial in the form $\boldsymbol{c}_1 + \boldsymbol{c}_2 X$, so the result is a degree two polynomial, containing three

components. One issue that makes multiplications displeasing in FHE is that the dimension of the resulting ciphertext becomes larger. In principle, it could be possible to decrypt it, using $sk \cdot sk$ as the new secret. Nevertheless, this quickly becomes unpractical.

3.2.1 Key switching

One of the most important features of BGV, firstly introduced by Brakerski and Vaikuntanathan in [BV11], is the so-called *key switching*. Initially, it has been introduced as a way to reduce the size of the ciphertext after a multiplication (it was originally referred to as *dimensionality reduction*). Sometimes it is referred to as *relinearization*. Currently, it is the only known approach to make BGV-like schemes compact. We first give an idea of the procedure based on LWE, so that the intuition is easier.

[BV11]: Brakerski and Vaikuntanathan, *'Efficient Fully Homomorphic Encryption from (Standard) LWE'* . In: FOCS '11

An explanation in plain LWE terms. Since under RLWE the interpretation is a bit more difficult, we first give an intuition considering the Regev scheme (Section 2.3.1). Note that under LWE we use the term $\boldsymbol{x}$ to refer to sk. Consider a setting composed of a public key $B = (\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{s} \mid A) \in \mathbb{Z}_q^{m \times n}$, where $\boldsymbol{b} \in \mathbb{Z}_q^m$ is a LWE sample and $A \in \mathbb{Z}_q^{m \times (n-1)}$ is uniformly distributed. We know by Regev scheme that a ciphertext is in the shape $\boldsymbol{c} := A^t\boldsymbol{r} + (q/2 \cdot \mu, 0, \ldots, 0) \in \mathbb{Z}_q^n$ – let us ignore the particular encoding of the message.

After a multiplication (which, in LWE terms, is evaluated as a tensor product), the size of the ciphertext grows from n to n^2. Still, the ciphertext is decryptable using $\boldsymbol{x} \otimes \boldsymbol{x}$:

$$[\langle \boldsymbol{c} \otimes \boldsymbol{c}', \boldsymbol{x} \otimes \boldsymbol{x} \rangle]_q \equiv [\langle \boldsymbol{c}, \boldsymbol{x} \rangle] \cdot [\langle \boldsymbol{c}', \boldsymbol{x} \rangle] \in \mathbb{Z}_q$$

of course the size of $\boldsymbol{c}_{\text{mul}} = \boldsymbol{c} \otimes \boldsymbol{c}' \in \mathbb{Z}_q^{n^2}$ is too large. The key switching procedure firstly produces a new public key A_{ks} of size $n^2 \times n$, injected with a new (smaller) secret $\boldsymbol{x}' \in \mathbb{Z}_q^n$ following the usual LWE encryption. A little bit more formally:

$$A_{\text{ks}} = (\boldsymbol{b}' = A'\boldsymbol{x}' + \boldsymbol{s}' \mid A') \in \mathbb{Z}_q^{n^2 \times n}$$

where $A' \in \mathbb{Z}_q^{n^2 \times n-1}$ is uniformly distributed. At this point, the "old" secret key $\boldsymbol{x}$ is added to the first column of A_{ks},

resulting in the *key switching key* $\tau_{(\boldsymbol{x}\otimes\boldsymbol{x})\to\boldsymbol{x}'}$:

$$\tau_{(\boldsymbol{x}\otimes\boldsymbol{x})\to\boldsymbol{x}'} = (\boldsymbol{x}\otimes\boldsymbol{x} + \boldsymbol{b}' \mid A') \in \mathbb{Z}_q^{n^2\times n}.$$

Before justifying how $\tau_{(\boldsymbol{x}\otimes\boldsymbol{x})\to\boldsymbol{x}'}$ can be used to switch the key, we introduce two functions $\text{Bit}(\boldsymbol{a})$ and $\text{Pow}(\boldsymbol{a})$.

Definition 58 (*Bit decomposition*)

The $\text{Bits}(\boldsymbol{a})$ *function performs the bit decomposition of a vector* $\boldsymbol{a}\in\mathbb{Z}_q$ *by decomposing each coefficient* a_i *into a binary vector of* $\log(q)$ *elements. In practice, given*

$$\text{bits}(x) = (x_1,\ldots,x_n) \quad \text{s.t.} \quad x = \sum_{i=1}^{n} x_i\, 2^i, \quad x_i\in\{0,1\},$$

we define $\text{Bits}(\boldsymbol{a})$ *as the application of* $\text{bits}(x)$ *on every* a_i*:*

$$\text{Bits}(\boldsymbol{a}) = (\text{bits}(a_1),\text{bits}(a_2),\ldots,\text{bits}(a_n)) \in \{0,1\}^{n\log(q)}$$

Definition 59 (*Powers of two*)

The $\text{Pow}(\boldsymbol{b})$ *function reconstructs a vector* $\boldsymbol{a}\in\mathbb{Z}_q^n$ *from its bit decomposition* $\boldsymbol{b}\in\{0,1\}^{n\cdot\log(q)}$*. Given*

$$\text{pow}((b_0,\ldots,b_{n-1})) = \sum_{i=0}^{n-1} b_i\, 2^i,$$

we define $\text{Pow}(\boldsymbol{b})$ *as the application of* pow *to every row of* $\boldsymbol{b}$*:*

$$\text{Pow}(\boldsymbol{b}) = (\text{pow}(\boldsymbol{b}_0),\text{pow}(\boldsymbol{b}_1),\ldots,\text{pow}(\boldsymbol{b}_{n-1})) \in \mathbb{Z}_q^n.$$

Clearly, $\text{Pow}(\text{Bits}(\boldsymbol{a})) = \boldsymbol{a}$ and, surprisingly,

$$\langle \text{Bits}(\boldsymbol{a}), \text{Pow}(\boldsymbol{b})\rangle = \langle \boldsymbol{a},\boldsymbol{b}\rangle, \tag{3.3}$$

so we can transform some vector $\boldsymbol{a}$ to a (longer) binary vector and maintaining its inner product with $\boldsymbol{b}$.

At this point, by simply evaluating $\text{Bits}(\boldsymbol{c}_{\text{mul}})^t\cdot\tau_{(\text{Pow}(\boldsymbol{x}\otimes\boldsymbol{x}))\to\boldsymbol{x}'}$, we obtain a new ciphertext decrypting to the same message

as $\boldsymbol{c}_{\text{mul}}$ but under the new key $\boldsymbol{x}'$.

$$\begin{aligned}
\text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot \tau_{(\boldsymbol{x}\otimes\boldsymbol{x})\to\boldsymbol{x}'} &= \text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot (\text{Pow}(\boldsymbol{x}\otimes\boldsymbol{x}) + \boldsymbol{b}' \mid A') \\
&= (\langle \text{Bits}(\boldsymbol{c}_{\text{mul}}), \text{Pow}(\boldsymbol{x}\otimes\boldsymbol{x})\rangle + \langle \text{Bits}(\boldsymbol{c}_{\text{mul}}), \boldsymbol{b}'\rangle \mid \text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot A') \\
&= (\langle \boldsymbol{c}_{\text{mul}}, \boldsymbol{x}\otimes\boldsymbol{x}\rangle + \langle \text{Bits}(\boldsymbol{c}_{\text{mul}}), \boldsymbol{b}'\rangle \mid \text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot A') \\
&= (m + \langle \text{Bits}(\boldsymbol{c}_{\text{mul}}), \boldsymbol{b}'\rangle \mid \text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot A') \\
&= (A_{\text{ks}})^t \cdot \text{Bits}(\boldsymbol{c}_{\text{mul}}) + (m, 0, \ldots, 0) \in \mathbb{Z}_q^n
\end{aligned}$$

where m is the decryption of the multiplication $m = \langle \boldsymbol{c}_{\text{mul}}, \boldsymbol{x}\otimes\boldsymbol{x}\rangle$ (without error removal). Observe that, since a ciphertext is uniformly distributed by definition, then $\text{Bits}(\boldsymbol{c}_{\text{mul}})$ acts as the random noise $\boldsymbol{r} \in \{0,1\}^m$ sampled during the encryption. In other words, the previous product has the same "meaning" as the classical encryption $A^t\boldsymbol{r}+(m,0,\ldots,0)$, with the difference that message is already noisy. Unluckily, its level of noise is also increased as the effect of adding the "new random mask" $\text{Bits}(\boldsymbol{c}_{\text{mul}})^t \cdot A_{\text{ks}}$ to it. Of course, the new (short) ciphertext can be decrypted using the new secret key $\boldsymbol{x}$. Theoretically, we would have to define a set of keys for each level ℓ, since after a multiplication the key is changed. By admitting circular security [MV24], that is the encryption of a secret key under the corresponding public key, we can switch back to the original key so that the number of keys does not depend on the circuit anymore.

[MV24]: Micciancio and Vaikuntanathan, *'SoK: Learning with Errors, Circular Security, and Fully Homomorphic Encryption'* . In: PKC '24

Notice that this procedure can be generalized to switch the key from any dimension n_1 to n_2, but we will focalize on the case of switching from n^2 to n. In the case of RLWE, this corresponds to switch from ciphertexts having three components to two components, as it will be shown below.

Bridging to RLWE. As in the previous case, the actual RLWE key switching procedure is composed of two phases: the first one produces all the public material required to switch the key and is performed by the key owner. The second one is run by the server running the encrypted computation that does not have access to the secret key. Notice that now the notation for the secret key is $sk = \boldsymbol{s}$.

We define two ways of performing this operation. The first one is referred to as BV, following its first introduction [BV11]. The idea, to keep the noise growth small, is to decompose the ciphertext, like in the previously presented case using $\text{Bits}(\boldsymbol{a})$ and $\text{Pow}(\boldsymbol{a})$. We generalize that argument to decompose with respect to any base ω, and not necessarily $\omega = 2$ like in the previous case. Following the notation from [KPZ21]. Let

[KPZ21]: A. Kim, Polyakov, and Zucca, *'Revisiting Homomorphic Encryption Schemes for Finite Fields'* . In: ASIACRYPT '21

$\ell_{\omega,Q} = \lfloor \log_\omega(Q) \rceil + 1$, we define, for any $\boldsymbol{a} \in \mathcal{R}$, function $\mathcal{D}$ and $\mathcal{P}$ as follows:

$$\begin{aligned} \mathcal{D}_{\beta,q_\ell}(\boldsymbol{a}) &= ([\boldsymbol{a}]_\omega, [\lfloor \frac{\boldsymbol{a}}{\omega} \rceil]_\omega, \ldots, [\lfloor \frac{\boldsymbol{a}}{\omega^{(\ell_{\omega,q_\ell})-1}} \rceil_\omega) \in \mathcal{R}_\omega^{\ell_\omega, q_\ell} \\ \mathcal{P}_{\beta,q_\ell}(\boldsymbol{a}) &= ([\boldsymbol{a}]_{q_\ell}, [\boldsymbol{a}\omega]_{q_\ell}, \ldots, [\boldsymbol{a}\omega^{(\ell_{\omega,q_\ell})-1}]_\omega) \in \mathcal{R}_{q_\ell}^{\ell_\omega, q_\ell} \end{aligned} \tag{3.4}$$

The first function, $\mathcal{D}$, decomposes the coefficients of a polynomial in radix base ω, and the second $\mathcal{P}$ retrieves the powers of ω. By combining them, we can recover the original polynomial $\boldsymbol{a}$. Like in the case of $\text{Bits}(\boldsymbol{x})$ and $\text{Pow}(\boldsymbol{x})$ – see Equation 3.3 – the product operation can be maintained after the application of these transformations, namely

$$\langle \mathcal{D}_{\omega,Q}(\boldsymbol{a}), \mathcal{P}_{\omega,Q}(\boldsymbol{b}) \rangle \equiv \langle \boldsymbol{a}, \boldsymbol{b} \rangle \bmod Q.$$

We therefore introduce the first phase, that aims to generate the public material required to perform this operation.

KeySwitchGen$^{\text{BV}}(\boldsymbol{s}, \boldsymbol{s}')$

Given as input a "starting" secret key $\boldsymbol{s}$ and a target secret key $\boldsymbol{s}'$, sample $\boldsymbol{a}_{\text{ks}} \in \mathcal{R}_{q_L}$ uniformly and $\boldsymbol{e}_{\text{ks}} \sim \chi_{\text{err}}$. Define

$$ks^{\text{BV}} := (\boldsymbol{a}_{\text{ks}} \cdot \boldsymbol{s}' + t\boldsymbol{e} + \mathcal{P}_{\omega,q_\ell}(\boldsymbol{s}), -\boldsymbol{a}_{\text{ks}}) \in \mathcal{R}_{q_\ell}^2$$

Notice that this key is added to the evaluation keys evk. And the actual operation is performed on an "extended" ciphertext as follows.

KeySwitch$^{\text{BV}}(ks^{\text{BV}}, \boldsymbol{c})$

Given a ciphertext $\boldsymbol{c}$ under some key $\boldsymbol{s}$ and modulus q_ℓ, switch key to $\boldsymbol{s}'$ as

$$\begin{aligned} \boldsymbol{c}' := ([\boldsymbol{c}_1 + \langle \mathcal{D}_{\omega,q_\ell}(\boldsymbol{c}_2), \boldsymbol{ks}_1^{\text{BV}} \rangle)]_{q_\ell}, \\ [\boldsymbol{c}_2 + \langle \mathcal{D}_{\omega,q_\ell}(\boldsymbol{c}_3), \boldsymbol{ks}_2^{\text{BV}} \rangle]_{q_\ell}) \in \mathcal{R}_{q_\ell}^2 \end{aligned}$$

See [KPZ21, Section B.1.1] for a proof of correctness of this phase. The second type of key switching was proposed by Gentry, Halevi and Smart [GHS12b]. The idea is, instead of decomposing the ciphertext, to temporarily extend the size of the modulus q with another modulus p and shifting the old key $\boldsymbol{s}$ by p, instead of evaluating the expensive $\mathcal{D}$ and $\mathcal{P}$ functions.

[GHS12b]: Gentry, Halevi, and Smart, *'Homomorphic Evaluation of the AES Circuit'* . In: CRYPTO '12

KeySwitchGen$^{\text{GHS}}(\boldsymbol{s}, \boldsymbol{s}')$

Given as input a "starting" secret key $\boldsymbol{s}$ and a target secret key $\boldsymbol{s}'$, sample $\boldsymbol{a}_{\text{ks}} \in \mathcal{R}_{q_L}$ uniformly and $\boldsymbol{e}_{\text{ks}} \sim \chi_{\text{err}}$. Define

$$ks^{\text{BV}} := (\boldsymbol{a}_{\text{ks}} \cdot \boldsymbol{s}' + t\boldsymbol{e} + p\boldsymbol{s}, -\boldsymbol{a}_{\text{ks}}) \in \mathcal{R}^2_{pq_L}.$$

Eventually, the key switching is performed is performed as follows.

KeySwitch$^{\text{GHS}}(ks^{\text{GHS}}, \boldsymbol{c})$

Given a ciphertext $\boldsymbol{c}$ under some key $\boldsymbol{s}$ and modulus q_i, switch key to $\boldsymbol{s}'$ as

$$\boldsymbol{c}' := (p\boldsymbol{c}_1 + \boldsymbol{c}_3 \cdot \boldsymbol{ks}_1^{\text{GHS}}, P\boldsymbol{c}_2 + \boldsymbol{c}_3 \cdot \boldsymbol{ks}_2^{\text{GHS}}) \in \mathcal{R}^2_{pq_\ell}.$$

In order to reduce the modulus of the new ciphertext, fix $\boldsymbol{\delta} = t[-\boldsymbol{c}'t^{-1}]_P$, with t^{-1} being the inverse modulo p, evaluate:

$$\boldsymbol{c}'' := \frac{\boldsymbol{c}' + \boldsymbol{\delta}}{p} \in \mathcal{R}^2_{q_i}$$

We will discuss about the modulus switching below. We refer the interested reader to [KPZ21] for more details about the proofs of correctness of the just presented procedure and about the noise growth during the key switching phase(s). Usually, to take the best from the two worlds, an hybrid approach is taken. Namely, there is a small increase in the modulus Q and also a decomposition with respect to a radix δ (see Section 3.1 of [GHS12b])

3.2.2 Modulus switching and the moduli ladder.

One of the most important features of BGV, which was not well explained above, is about the flexibility of the modulus. Let us first introduce the problem of noise in multiplications: consider a product $\boldsymbol{c}_{\text{mul}} = \boldsymbol{c} \cdot \boldsymbol{c}'$. Given that both $\boldsymbol{c}, \boldsymbol{c}'$ are fresh encryptions, it can easily be shown that the amount of noise in $\boldsymbol{c}_{\text{mul}}$ is roughly squared. Say that the noise goes from $\|\boldsymbol{v}\|_\infty$ to $\|\boldsymbol{v}\|^2_\infty$. Then, after another multiplication, it goes from $\|\boldsymbol{v}\|^2_\infty$ to $\|\boldsymbol{v}\|^4_\infty$, and so on. It is easy to see that this growth is exponential and not practical.

It turns out that, by simply scaling the ciphertext from one modulus q to another $p < q$, then also the amount of noise gets reduced. This is exactly the solution proposed in [BGV12], that is to perform a *modulus switching* operation. We report

a Lemma that gives an intuition of this result in LWE terms, without loss of generality, in the case of binary plaintext space.

Lemma 22 ([BGV12])

Let p and q be two odd moduli, and let $\boldsymbol{c}$ be an integer vector. Define $\boldsymbol{c}'$ to be the integer vector closest to $(p/q)\cdot\boldsymbol{c}$ such that $\boldsymbol{c}' \equiv \boldsymbol{c} \bmod 2$. Then, for any $\boldsymbol{s}$ with $|\langle \boldsymbol{c}, \boldsymbol{s}\rangle|_q < q/2 - (q/p)\cdot \ell_1(\boldsymbol{s})$, we have

$$[\langle \boldsymbol{c}', \boldsymbol{s}\rangle]_p \equiv [\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_q \bmod 2 \quad \textit{and}$$
$$\left|[\langle [\boldsymbol{c}', \boldsymbol{s}\rangle]_p\right| < (p/q)\cdot \left|[\langle \mathbf{c}, \mathbf{s}\rangle]_q\right| + \ell_1(\mathbf{s}),$$

where $\ell_1(\boldsymbol{s})$ is the ℓ_1-norm of $\boldsymbol{s}$.

We give a visual intuition of this process in Figure 59

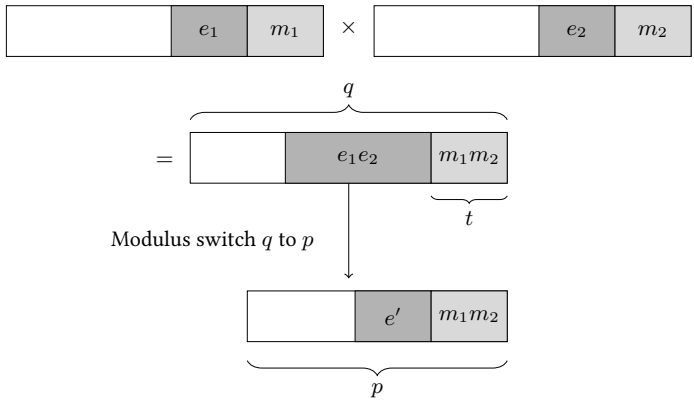


Figure 59: Representation of modulus switching after a BGV multiplication

One might argue though that this only changes the absolute values of noise, but its relative value with respect to the modulus does not decrease at all. This is true, but in reality we are not interested in the amount of noise *with respect to the noise*, but to its absolute value. Consider the case where the modulus q is composed as a *ladder* of gradually decreasing moduli where $q_i \approx q/x^i$ for $i < k$. We saw that after a multiplication between two ciphertexts, we might get a level of noise of roughly $\|\boldsymbol{v}\|_\infty^2$. By scaling the $\bmod q$ ciphertext by $1/x$, we have by Lemma 22 that the noise with respect to q/x can be reduced back to $\|\boldsymbol{v}\|_\infty$ (ignoring the impact of $\ell_1(\boldsymbol{s})$). At this point, by performing a multiplication between two ciphertexts at the same "level" – i.e., both under q/x – the noise is again increased to $\|\boldsymbol{v}\|_\infty^2$, but the same technique can be used again to switch the modulus to q/x^2 and refreshing the amount of noise.

With this approach, it is possible to evaluate up to k levels of multiplications, and not just $\log(k)$, as in the previous approach without modulus switching!

ModSwitch($\boldsymbol{c}, k$)

Given a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$ at level ℓ, integer $k \in \mathbb{N}$, set $\boldsymbol{\delta} = t \cdot [-\boldsymbol{c} \cdot t^{-1}]_{q_\ell / q_{\ell-k}}$, where t^{-1} is the inverse mod $q_\ell / q_{\ell-k}$, and set

$$\boldsymbol{c}' = \frac{q_{\ell-k}}{q_\ell}(\boldsymbol{c} + \boldsymbol{\delta}) \in \mathcal{R}_{q_{\ell-k}}^n,$$

output $\boldsymbol{c}'$.

Notice that the $\boldsymbol{\delta}$ term (i) does not influence the message, as it is such that $\boldsymbol{\delta} \equiv \mathbf{0} \bmod t$, and (ii) makes the ciphertext divisible by $q_\ell / q_{\ell-k}$, namely $\boldsymbol{\delta} + \boldsymbol{c} \equiv \mathbf{0} \bmod q_\ell / q_{\ell-k}$. The correctness of this procedure can be verified, refer to Section 3.3 of [BGV12]. We will give more intuitions on the lattice/geometrical meaning of this operation in Chapter 4.

We will not discuss about concrete way to pick BGV parameters, but we refer the reader to, e.g., [Mon+23].

For completeness, we also refer to the Brakerski/Fan-Vercauteren (BFV) scheme [Bra12; FV12], which can be seen as the case of BGV where the message is encoded as $m \cdot q/t$, instead of simply m. Since the message is stored in the most significant bits part of the ciphertext, no (explicit) modulus switching is required. On the other hand, the decryption algorithm is slightly different.

Nevertheless, it is possible to, given a ciphertext under BGV, scheme-switch it to BFV. In Appendix A of [AP13], they present a simple method to switch between the LSB and MSB encodings for RLWE ciphertexts. We give an intuition of BFV encoding and multiplication in Figure 60.

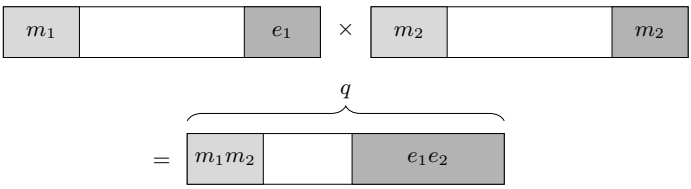


Figure 60: Representation of the resulting ciphertext after a BFV multiplication

Note that this different encoding requires the decryption algorithm to be tweaked accordingly. Lastly, we refer the interested reader to a generalization of BFV that allows to encode very large integer values [GV25].

[Bra12]: Brakerski, *'Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP'* . In: CRYPTO '12

[FV12]: Fan and Vercauteren (2012), Somewhat Practical Fully Homomorphic Encryption (Cryptology ePrint Archive, Paper 2012/144)

[AP13]: Alperin-Sheriff and Peikert, *'Practical Bootstrapping in Quasilinear Time'* . In: CRYPTO '13

[GV25]: Geelen and Vercauteren, *'Fully Homomorphic Encryption for Cyclotomic Prime Moduli'* . In: EUROCRYPT '25

Bootstrapping in BGV. As briefly mentioned in Section 1, the only trick that allows to obtain a (practical) fully homomorphic encryption scheme is the bootstrapping operation. The goal of this procedure is to homomorphically evaluate the decryption circuit. We will not discuss how this is performed – although we will do it for CKKS, since bootstrapping is required for the applications we will present – but we refer the interested reader to the general framework by Geelen and Vercauteren [GV23].

[GV23]: Geelen and Vercauteren (2023), *'Bootstrapping for BGV and BFV Revisited'* . In: Journal of Cryptology

3.2.3 Digression on SIMD operations.

In actual implementations, a plaintext polynomial can be seen as a vector of plaintext values from finite fields following the ring isomorphism firstly proposed in [SV10; SV14]. Take, for instance, the following set of parameters: $n = 4096, t = 65537$ (a Fermat prime) and $q = 2^{84}$. The cyclotomic polynomial is $\Phi_{2n} = \Phi_{8192}$. Let us consider the plaintext ring

$$\mathcal{R}_t = \mathbb{Z}_{65537}[X]/\Phi_{8192} = \mathbb{Z}_{65537}[X]/(X^{4096} + 1).$$

[SV10]: Smart and Vercauteren, *'Fully homomorphic encryption with relatively small key and ciphertext sizes'* . In: PKC '10
[SV14]: Smart and Vercauteren (2014), *'Fully homomorphic SIMD operations'* . In: Design, Codes and Cryptography

Since $65537 \equiv 1 \bmod 8192$, the polynomial $x^{4096} + 1$ splits completely modulo 65537 into 4096 linear factors.

Let $\alpha_1, \alpha_2, \ldots, \alpha_{4096} \in \mathbb{Z}_{65537}$ be the distinct roots. Then, the CRT gives the following isomorphism:

$$\begin{aligned} \phi : \mathcal{R} &\longrightarrow \underbrace{\mathbb{Z}_{65537} \times \cdots \times \mathbb{Z}_{65537}}_{\text{4096 times}} \\ a(x) &\longmapsto \big(a(\alpha_1), a(\alpha_2), \ldots, a(\alpha_{4096})\big) \end{aligned}$$

This isomorphism preserves addition and multiplication:

$$\begin{aligned} \phi(a(x) + b(x)) &= \phi(a(x)) + \phi(b(x)), \\ \phi(a(x) \cdot b(x)) &= \phi(a(x)) \cdot \phi(b(x)). \end{aligned}$$

Hence, each α_i corresponds to a SIMD slot, and operations on polynomials act component-wise on all slots. Remark that in an ideal lattice over a power of two cyclotomic polynomial, planting a short vector corresponds to planting many of them, since by the algebraic properties of the basis for an ideal lattice, given $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{L}$, then also $(-x_n, x_1, \ldots, x_{n-1}) \in \mathcal{L}$, but the ℓ_2 norm is the same (is just a rotation). A way of finding the (nega) rotation of some vector in the lattice, is simply by squaring it. In polynomial terms, this means that

each coefficient for each X^i is decoded using the i-th short vector.

3.3 Cheon-Kim-Kim-Song (CKKS)

A few years after the introduction of BGV, in 2017, Cheon, Kim, Kim and Song [CKKS17] presented the so-called Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) scheme, usually referred to as CKKS. The core idea about their scheme is to consider the error e, naturally present in encryptions for security reasons, as part of the message. Under this scheme, a message m is indeed decrypted to $m + e$. Technically, this does not satisfy the conditions of a public-key scheme, where $\text{Dec}(\text{Enc}(m)) = m$, indeed this holds only approximately as $\text{Dec}(\text{Enc}(m)) \approx m$. Nevertheless, as it will be shown later, this has many practical advantages. We will firstly give an informal presentation of the scheme, and then we will formally introduce all its building blocks. Also notice that more efficient instantiations of the scheme are available in the literature, e.g., starting from the first full RNS-based [Che+18a] to the error-reduced version [KPP22], although we describe for simplicity the original formulation of the scheme.

[CKKS17]: Cheon, A. Kim, M. Kim, and Song, *'Homomorphic Encryption for Arithmetic of Approximate Numbers'* . In: ASIACRYPT '17

[Che+18a]: Cheon, Han, A. Kim, M. Kim, et al., *'A Full RNS Variant of Approximate Homomorphic Encryption'* . In: SAC '18

[KPP22]: A. Kim, Papadimitriou, and Polyakov, *'Approximate Homomorphic Encryption with Reduced Approximation Error'* . In: CT-RSA '22

Some mathematical definitions. Let us fix some value $n \in \mathbb{Z}^+$, for simplicity we always assume $n = 2^k$ for some positive k – similar arguments can be done using arbitrary n and $\phi(n)$, but usually implementations stick to powers of two for efficiency. Remark Minkowksi embedding (Definition 44) and notice that its images belong to the subring $\mathbb{H} = \{(z_i)_{i \in \mathbb{Z}^*_{2n}} : z_i = \overline{z_{-i}}\}$ of $\mathbb{C}^n$, where $\mathbb{Z}^*_{2n}$ refers to $\mathbb{Z}^*_{2n} = \{x \in \mathbb{Z}_{2n} : \gcd(x, 2n) = 1\}$. For instance, $\mathbb{Z}^*_8 = \{1, 3, 5, 7\}$. In other words, $\mathbb{H}$ contains redundancy given by the fact that it contains elements plus their conjugates (see Theorem 6), and this happens as the roots of unity are symmetrically distributed on the unit circle (for a visual reference, see Figure 54). As an example, consider the roots of unity for $X^4 + 1$, namely ξ_8, ξ_8^3, plus their (reversed) conjugates, i.e., $\overline{\xi_8} = \xi_8^7$ and $\overline{\xi_8^3} = \xi_8^5$.

Lemma 23 (*The image of Minkowski embedding*)

The image of the Minkowski embedding $\sigma(\mathcal{R})$ is such that $\sigma(\mathcal{R}) \subseteq \mathbb{H} \cong \mathbb{C}^{n/2}$.

It is possible to see this with an example: consider the cyclotomic polynomial $X^4 + 1$, and observe that it is possible to decompose it over the reals as $(X^2 + \sqrt{2}X + 1)(X^2 - \sqrt{2} + 1)$, it is clear that the set of solution is given by $n/2$ values in $\mathbb{C}$ plus their conjugates. Since the conjugates does not add any new information, this generates the $\mathbb{C}^{n/2}$ space.

Lastly, we define the $\pi(\cdot)$ projection that computes:

$$\pi : \mathbb{H} \to \mathbb{C}^{n/2},$$

this simply takes an element in $\mathbb{H}$ and removes its last $n/2$ components, that are the conjugates of the first $n/2$. Conversely, $\pi^{-1}(\cdot)$ can be seen as an "expansion", where a vector in $\mathbb{C}^{n/2}$ gets expanded with its conjugates.

Encoding a set of complex numbers. The encoding process is somehow the inverse as the one performed in the NTT, but in the complex domain – it is indeed an inverse discrete Fourier transform. Consider a vector of complex values that we want to encode, for instance $\boldsymbol{a} = (2 + 3i, 0.4 - 6i) \in \mathbb{C}^2$. Expand it using the π^{-1} map as

$$\pi^{-1}(a(X)) = (2+3i)+(0.4-6i)X+(0.4+6i)X^2+(2-3i)X^3$$

After the expansion, this can be seen as an element of $\mathbb{H} \cong \mathbb{C}^2$. Since $\pi^{-1}(a(X))$ is in $\mathbb{H}$, it can be seen as the image of the σ embedding for some polynomial in $\mathbb{C}[X]/(X^n + 1)$. The objective now is to find some polynomial $b(X)$ such that $\sigma(b(X)) = \pi^{-1}(a(X))$.

Remark (e.g. from Example 8) that the Minkowski embedding can be evaluated by using the Vandermonde matrix, and its inverse σ^{-1} can be evaluated using the inverse matrix. In our example, with $n = 4$, the Vandermonde matrix using the 8-th primitive root of unity $\xi_8 = e^{\pi i/8}$ corresponds to:

$$V = \begin{pmatrix} 1 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i & i & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i \\ 1 & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i & -i & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i \\ 1 & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i & i & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i \\ 1 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i & -i & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i \end{pmatrix} \in \mathbb{C}^{4\times 4}.$$

Therefore, we could define $\sigma^{-1}(a(X)) = V^{-1} \cdot \boldsymbol{a}$. By doing so, we find that the polynomial $b(X)$ such that $\sigma(b(X)) = \pi^{-1}(a(X))$ is

$$b(X) = 1.2 - 0.49X + 4.5X^2 - 1.62X^3 \in \mathbb{R}[X](X^4 + 1)$$

On the other hand, by evaluating $b(X)$ in four the roots of unity $\xi_8, \xi_8^3, \xi_8^5, \xi_8^7$ we obtain precisely $\pi^{-1}(a(X))$ – this will be our decoding procedure. However, we have no guarantees that the result of this mapping will be an integer polynomial, indeed in the above case $b(X)$ has real coefficients. In other words, there are some polynomials in $\mathbb{C}[X]/(X^n+1)$ that are not images of $\sigma(p(X))$ for any $p(X) \in \mathbb{Z}[X]/(X^n+1)$

Nonetheless, in order to base the security of the scheme on RLWE – and to have quick integer computations – we need to encode $b(X)$ as an integer polynomial. Luckily, we can use a trick called "coordinate-wise random rounding" presented in [LPR13] that allows to round a polynomial to the closest integer point in the lattice defined by $\sigma(\mathcal{R})$.

[LPR13]: Lyubashevsky, Peikert, and Regev, *'A Toolkit for Ring-LWE Cryptography'* . In: EUROCRYPT '13

The idea is as follows: remark that $\mathbb{Z}[X]/(X^n+1)$ has an orthogonal basis $(1, X, \ldots, X^{n-1})$. Since the Minkowski embedding σ is an homomorphism, also $(\sigma(1), \sigma(X), \ldots, \sigma(X^{n-1})$ has an orthogonal basis. Geometrically, the Minkowski embedding defines an isometry (i.e. a scaled rotation) of the $\mathbb{Z}^n$ lattice[1]. As such, and since we know that the orthogonal basis for the lattice generated by $\sigma(\mathcal{R})$ is simply the Vandermonde matrix (refer to Section 2.4.1.3), and the basis is orthogonal, we can round the real valued point with respect to the lattice using the basis V. We will refer to this procedure as $\lfloor \boldsymbol{a} \rceil_{\sigma(\mathcal{R})}$, and it works as follows:

1: Two lattices that differ by a rotation and/or a scaling are isomorphic

1. Given $\pi^{-1}(a(X)) \in \mathbb{C}^{n/2}$, find the set of coefficients $\boldsymbol{x} \in \mathbb{R}^n$ of $a(X)$ in the underlying orthogonal lattice defined by $\sigma(\mathcal{R})$. To do so, we can use the basis V and project $\boldsymbol{x}$ along all the $i < n$ dimensions as $\langle \boldsymbol{x}, \boldsymbol{v}_i \rangle / \langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle$
2. Randomly round the coefficients, i.e. evaluate $x_i \rightarrow \lfloor x_i + e \rceil$ for uniformly random $e \in \{-1, 0, 1\}$
3. Return the point as $V \cdot \boldsymbol{x}$ using the new $\boldsymbol{x}$.

We can visualize this procedure as shown in Figure 61.

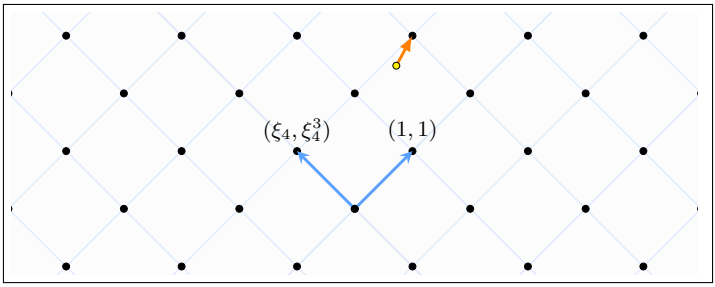


Figure 61: Rounding a point in the rotated $\mathbb{Z}^2$ lattice generated by the two basis vectors $(1, 1)$ and (ξ_4, ξ_4^3), namely the columns of the Vandermonde matrix generated by the roots of $X^2 + 1$

Notice that rounding can be done easily as the lattice has a known orthogonal basis. The reader should not confuse the

rotated $\mathbb{Z}^n$ lattice (which acts as a "ambient space", we will discuss more about this concept in Chapter 4) and the actual random q-ary lattice that is constructed inside it (remark (2.14)) It is hard to decode the target with respect to the latter, not to the former.

For more details and justifications about this procedure, we refer to Section 2.4.2 of [LPR13].

Notice that the least significant bits of the message might be destroyed by this procedure, therefore the message is scaled by a so-called scaling factor $\Delta > 1$ before the rounding.

The complete encoding procedure is eventually evaluated by computing the σ^{-1} transformation over the polynomial:

$$\text{Encode}(\boldsymbol{a}) := \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{a}) \rceil_{\sigma(\mathcal{R})}\right)$$

This process generates the object that we were looking for at the beginning of the paragraph, namely a polynomial $b(X) \in \mathbb{Z}[X]/(X^n+1)$ such that, when evaluated in the roots of unity, returns (approximately) the message to be encoded. Notice that we will also refer to this procedure as the inverse discrete Fourier transform (IDFT)

The decoding procedure is simply evaluated like that, therefore:

$$\text{Decode}(\boldsymbol{a}) := \pi(\sigma(\boldsymbol{a} \cdot \Delta^{-1})),$$

and it is referred to as the discrete Fourier transform (DFT). Notice that, by the ring homomorphism in (2.25), we have that by multiplying two encoded polynomials, we obtain the slot-wise multiplication of the underlying messages, leading to efficient SIMD computations.

Example 12 (*Encoding/decoding in CKKS*)

Given $n = 2$ and $\mathcal{R} = \mathbb{Z}[X](X^2+1)$, we want to encode a message containing $n/2$ complex numbers. We set $\Delta = 64$, and let us pick

$$\boldsymbol{m} = (5 + 0.4i) \in \mathbb{C}^{n/2}.$$

The first step is the expansion:

$$\pi^{-1}(\boldsymbol{m}) = (5 + 0.40i, 5 - 0.40i) \in \mathbb{C}^n$$

and scale it by Δ:

$$\Delta \cdot \pi^{-1}(\boldsymbol{m}) = (320 + 25.60i, 320 - 25.60i) \in \mathbb{C}^n$$

At this point we have an element of $\mathbb{H}$, and we round it

using the coordinate-wise rounding to obtain a lattice point in $\sigma(\mathcal{R})$

$$\lfloor(\pi^{-1}(\boldsymbol{m}))\rfloor_{\sigma(\mathcal{R})} = (320 + 26i, 320 - 26i)$$

So now we have obtained a lattice point, which can be transformed using the inverse σ^{-1} to an integer preimage of $\mathbb{H}$ as

$$\begin{aligned}\sigma^{-1}(\lfloor(\pi^{-1}(\boldsymbol{m}))\rfloor_{\sigma(\mathcal{R})}) &= V^{-1} \cdot \lfloor(\pi^{-1}(\boldsymbol{m}))\rfloor_{\sigma(\mathcal{R})} \\ &= V^{-1} \cdot (320 + 26i, 320 - 26i) \\ &= (320, 26) \in \mathbb{Z}^2\end{aligned}$$

where V is the Vandermonde matrix defined as

$$V = \begin{pmatrix} 1 & \xi_4 \\ 1 & \xi_4^3 \end{pmatrix} = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \in \mathbb{C}^{2\times 2},$$

We therefore obtained the encoding of the message, which is the polynomial

$$\text{encode}(\boldsymbol{m}) = 320 + 26X \in \mathcal{R}$$

Theoretically, this polynomial is compatible with RLWE encryption and can be easily be encrypted following, e.g., encryption in BGV (with small modifications). We now show how the decoding process simply inverts the encoding and returns an approximate version of $\boldsymbol{m}$. Invert the $\sigma(\mathcal{R})$ embedding using the V matrix:

$$V \cdot (320, 26) = (320 + 26i, 320 - 26i) \in \mathbb{C}^n,$$

and scale down the using $1/\Delta$:

$$1/64 \cdot (320 + 26i, 320 - 26i) = (5.0 + 0.41i, 5.0 - 0.41i).$$

As a last step, use the π map to obtain the decoding:

$$\pi((5.0 + 0.41i, 5.0 - 0.41i)) = (5 + 0.41i) \in \mathbb{C}^{n/2}.$$

Observe that there is a small loss in the precision of the message (i.e., we obtain $\boldsymbol{m} + (0, 0.1i)$), which is normal and can be controlled using larger values of Δ (at a cost).

The concrete scheme. The setup and key generation phases of CKKS are very similar to the ones of BGV. We first de-

fine some distributions, adopting the notation in [CKKS17; GHS12b]:

- $\mathcal{ZO}(\rho)$: for a real $0 \leq \rho \leq$, this distribution draw one entry from $\{-1, 0, 1\}^n$, with probability $\rho/2$ for each $-1, 1$ and $1 - \rho$ for the value 0.
- $\mathcal{HWT}(h)$: for a positive integer h, this distribution draws an entry from $\{-1, 0, 1\}^n$ having Hamming weight equal to $h > 0$
- $\mathcal{DG}(\sigma)$: a discrete Gaussian distribution with parameter $\sigma > 0$ (refer to Section 2.1.6 for details about discrete Gaussians).

Setup

Given a scaling factor $\Delta > 0$, fix a modulus q_1 and let $q_\ell \approx \Delta^\ell \cdot q_1$ for $0 < \ell \leq L$, where $L > 0$ is the number of evaluable levels. Fix a parameter $n := n(\lambda, q_L)$ that is a power of two, the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\Phi_n(X)$ and an Hamming weight $h = h(\lambda, q_L)$

Notice that each element of the moduli chain differs approximately by Δ. Key generation follows from RLWE.

Keygen(λ)

For the secret key, sample $\boldsymbol{s} \sim \mathcal{HWT}(h)^n$ and set:

$$sk := \boldsymbol{s} \in \mathcal{R}_{q_L},$$

Then sample $\boldsymbol{a} \sim \mathcal{R}_{q_L}$ uniformly and $\boldsymbol{e} \leftarrow \mathcal{DG}(\sigma)$, set:

$$pk := ([\boldsymbol{a} \cdot \boldsymbol{s} + t\boldsymbol{e}]_q, -\boldsymbol{a}) \in \mathcal{R}_{q_L}^2.$$

Return (pk, sk).

Given an encoded polynomial in $\mathbb{Z}[X]/(X^n + 1)$, we know how to efficiently encrypt it using RLWE, simply follow the standard methods from, e.g., the BGV scheme. Observe that the key generation is shared among the different schemes, and their difference is simply about how to encode the message. Also observe that in BGV-like schemes, SIMD computations can be performed by picking some well-chosen set of parameters and by taking advantage of the CRT decomposition of the plaintext ring modulo Φ_{2n} over $\mathbb{Z}_t$ with $t \equiv 1 \bmod n$. On the other hand, in CKKS, we exploit the Minkowski embedding – in fact, that we do not really talk about a plaintext modulus in CKKS, as the scheme is approximate and would not make sense. An intuition is that in BGV slots are *algebraic*, while in CKKS they are *geometrical*. We refer to Appendix C of [GHS12a] for more information.

[GHS12a]: Gentry, Halevi, and Smart, *'Fully Homomorphic Encryption with Polylog Overhead'* . In: EUROCRYPT '12

$\mathbf{Ecd}_{\Delta}(\boldsymbol{m})$

Given a complex vector $\boldsymbol{m} \in \mathbb{C}^{n/2}$, return

$$p(X) := \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{m}) \rceil_{\sigma(\mathcal{R})}\right) \in \mathcal{R}.$$

Encryption is performed following RLWE and is almost the same as in BGV, except for the usage of the plaintext modulus t.

$\mathbf{Enc}_{pk}(p(X))$

Given a (encoded) polynomial $p(X) \in \mathcal{R}$, sample $\boldsymbol{u} \leftarrow \mathcal{ZO}(0.5)$ and $\boldsymbol{e}_1, \boldsymbol{e}_2 \sim \mathcal{DG}(\sigma)$. Return

$$\boldsymbol{c} := (\boldsymbol{m} + \boldsymbol{u} \cdot pk_1 + \boldsymbol{e}_1, \boldsymbol{u} \cdot pk_2 + \boldsymbol{e}_2) \in \mathcal{R}_{q_\ell}^2.$$

Similarly, we define decryption.

$\mathbf{Dec}_{sk}(\boldsymbol{c})$

Given the ciphertext $\boldsymbol{c} = \boldsymbol{c}_1 + \boldsymbol{c}_2 X$ and the secret key sk, evaluate $\boldsymbol{c}$ at $X = sk = \boldsymbol{s}$:

$$\begin{aligned} \boldsymbol{c}(sk) &\equiv [\boldsymbol{p} + \boldsymbol{u} \cdot pk_1 + t\boldsymbol{e}_1 + \boldsymbol{u} \cdot pk_2 \boldsymbol{s} + \boldsymbol{e}_2 \cdot \boldsymbol{s}]_q \\ &\equiv [\boldsymbol{p} + \boldsymbol{u} \cdot \boldsymbol{e} + \boldsymbol{e}_1 + \boldsymbol{e}_2 \cdot \boldsymbol{s})]_q \\ &\equiv \boldsymbol{p} + \boldsymbol{v} \in \mathcal{R}_{q_\ell} \end{aligned}$$

where we call $\boldsymbol{v}$ the amount of noise.

It is clear that this is not a discretized scheme, but only approximate, as the "original" message $\boldsymbol{p}$ cannot be recovered since it is masked by some random noise that can not be removed.

Lastly, to recover the actual message, we also have to decode the polynomial defined by $\boldsymbol{p}$.

$\mathbf{Dcd}_{\Delta}(\boldsymbol{m})$

Given a plaintext $p(X) \in \mathbb{Z}[X]/(X^n + 1)$ return:

$$\pi(\sigma(\boldsymbol{a} \cdot \Delta^{-1})) \in \mathbb{C}^{n/2}$$

that contains $\approx \boldsymbol{m}$.

Rescaling. In the case of BGV, after a multiplication, a modulus switching is performed to reduce the amount of noise in the resulting ciphertext. Recall that, in standard computers, when two floating/fixed-point numbers are multiplied, the error in the LSBs is rounded – intuitively, the multiplication of, e.g., two 8-bits numbers requires 16-bits to keep full precision.

Similarly, the role of modulus switching in CKKS, where the error is part of the message, can be seen as the same as rounding a large number by "cutting" its LSBs. Since a message is scaled by Δ, a multiplication will be scaled by Δ^2. Since the moduli chain is set such that each $q_\ell \approx \Delta^\ell \cdot q_1$, after a multiplication we scale the ciphertext by Δ to bring the scale back to the original one, truncating the LSBs. Of course the intrinsic limit on multiplications is still there, namely we can perform up to L multiplications. Differently from BGV, after L levels, we can not multiply anymore as there is no "modulus left" to cut. Under CKKS, we call this step *rescaling*; the syntax is similar to the modulus switching (although there is no $\boldsymbol{\delta}$ term here, as there is no discretized plaintext space), but the semantic is very different.

Rescale$_\Delta(\boldsymbol{c}, \ell)$

Given a ciphertext $\boldsymbol{c} \in \mathcal{R}^2_{q_\ell}$, return

$$\lfloor \frac{q_{\ell-1}}{q_\ell} \cdot \boldsymbol{c} \rceil = \lfloor \Delta^{-1} \cdot \boldsymbol{c} \rceil \bmod q_{\ell-1},$$

since $q_\ell = \Delta^\ell \cdot q_1$.

We remark that, since very large numbers are involved when using CKKS, all the standard implementations [KPP22] are based on DCRT polynomials, and sometimes referred to as RNS-CKKS. To conclude, CKKS implements a key switching algorithm similar to the ones presented for BGV – namely BV, GHS and Hybrid – to reduce the size of a ciphertext after a multiplication. This phase is sometimes called relinearization as the degree two ciphertext is reduced to a degree one polynomial. Additions and multiplications are performed similarly to BGV: namely additions are direct polynomial additions, and multiplications are followed by a rescaling operation and a key switching to set the key back to the original one. We omit details.

Rotations. The concept of rotating plaintext slots firstly appeared in [BGV12; GHS12a]. The high-level idea is, given a ciphertext containing an encryption of $m_1, m_2, \ldots, m_k$ packed elements, to obtain a valid encryption of, e.g., $m_k, m_1, \ldots, m_{k-1}$.

Definition 60 (*Galois group of a Cyclotomic Field*)

Let $n \in \mathbb{Z}_{>0}$ and let ξ_n be a primitive n-th root of unity. The cyclotomic field *is the number field $\mathbb{Q}(\xi_n)$.*
Its Galois group over $\mathbb{Q}$ consists of all field automorphisms that map

ξ_n to another primitive n-th root of unity:

$$\mathrm{Gal}(\mathbb{Q}(\xi_n)/\mathbb{Q}) = \{\, \kappa_k : \mathbb{Q}(\xi_n) \to \mathbb{Q}(\xi_n) \mid \kappa_k(\xi_n) = \xi_n^k,\ \gcd(k, n) = 1 \,\}.$$

The group operation is composition of automorphisms. Each automorphism κ_k corresponds to multiplication by k modulo n, giving a natural isomorphism

$$\mathrm{Gal}(\mathbb{Q}(\xi_n)/\mathbb{Q}) \ \cong\ (\mathbb{Z}/n\mathbb{Z})^{\times}.$$

In particular, the Galois group consists of all mappings

$$\kappa_k : p(X) \to p(X^k) \bmod \Phi_n(X).$$

Applying such a mapping has the effect of applying a permutation of the slots of the underlying plaintext, under the new secret key $\kappa_k(\boldsymbol{s})$. In our "vanilla" encoding for CKKS, we used the standard $\xi, \xi^3, \xi^5, \ldots, \xi^{n-1}$ representative of the $n/2$ pairs of conjugates to be evaluated in the decoding process. So, for instance, by simply using κ_3, the first slot will now refer to the second slot (i.e., $\xi \to \xi^3$), but the second will not do the same with the third (i.e. $\xi^3 \to \xi^6$, but the ξ^6 root does not encode any message). One solution, and the one used in modern implementations and proposals (e.g. see Section 4 of [Che+18b] or Section 2 of [CCS19]), is to use another set of representatives from ξ^{5^j}. This generates a group of order $n/2$, therefore by using this set of roots, we avoid the application of the π map during encoding and decoding. We use the following Vandermonde matrix:

[Che+18b]: Cheon, Han, A. Kim, M. Kim, et al., *'Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '18

[CCS19]: H. Chen, Chillotti, and Song, *'Improved Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '19

$$V = \begin{pmatrix} 1 & \xi_{2n}^1 & \xi_{2n}^2 & \cdots & \xi_{2n}^{n-1} \\ 1 & \xi_{2n}^5 & \xi_{2n}^{5\cdot 2} & \cdots & \xi_{2n}^{5\cdot(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_{2n}^{2n-3} & \xi_{2n}^{2\cdot(2n-3)} & \cdots & \xi_{2n}^{(2n-3)\cdot(n-1)} \end{pmatrix} \in \mathbb{C}^{n/2\times n},$$

Also notice that, for simplicity, we assume *full packed* ciphertexts, namely the ciphertexts contain the encryption of $n/2$ values. We could allow for less values (but always powers of two, e.g., $n/4$), thus enabling *sparse packing*, to speed up some procedures. For the sake of this explanation we avoid this technical optimization. Consider the following example.

Example 13 (*Rotation friendly representatives*)

Consider the eight roots of unity used to encode a message in

a polynomial in $\mathbb{Z}[X]/(X^8+1)$ that are

$$\xi_{16}^1, \xi_{16}^3, \xi_{16}^5, \xi_{16}^7, \xi_{16}^9, \xi_{16}^{11}, \xi_{16}^{13}, \xi_{16}^{15}$$

Now, observe that the amount of messages is $\mathbb{C}^{n/2} = \mathbb{C}^4$, so in principle we could recover the messages using the first four roots only.
Now, by instead picking $\xi_{16}^{5^i}$ as the primitive root (notice that is a generator of a multiplicative group $\bmod 2n$ of order $n/2$), we can obtain all the other roots by evaluating all the powers, as 5 is a generator. The set of roots used to encode/decode will therefore be

$$\xi_{16}^{5^0}, \xi_{16}^{5^1}, \xi_{16}^{5^2}, \xi_{16}^{5^3}$$

and this reduces to

$$\xi_{16}^1, \xi_{16}^5, \xi_{16}^9, \xi_{16}^{13}$$

Observe that $13^5 = 1 \bmod 16$, so we have a full cycle. By defining

$$\kappa_k : p(X) \to p(X^5) \bmod (X^{16}+1)$$

we can obtain rotations of the slots of the ciphertext by simply evaluating the κ_k map (and then by switching the key, as the new mapped ciphertext will be decryptable under $\kappa_k(\boldsymbol{s})$).

The reason why this set of roots is more convenient, is that now we can rotate the slots of a ciphertext $\boldsymbol{c}$ simply as $\kappa_k(\boldsymbol{c})$, with $k = 5^r$ and for some power $r < n$

The reason is that $\{1, 5, 9, \ldots, 2n-3\}$ is a cyclic multiplicative group $\bmod 2n$ of order $n/2$ generated by 5. By simply using this set of roots (that does not actually have any impact in the complexity of the encoding/decoding), we can unlock rotations for the underlying plaintext slots. Say that $\boldsymbol{m} \in \mathbb{C}^{n/2}$ is the cleartext, and it is encoded using the previously presented roots in $p(X)$, i.e., $p(\xi) = m_1, p(\xi^5) = m_2$ and so on.

Notice that the just presented procedure is an automorphism (hence an homomorphism) that should be applied to the secret key as well to obtain a correct decryption. Indeed, in order to correctly decrypt a ciphertext $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \in \mathcal{R}_q^2$ that was rotated via κ_k, we would need to decrypt it as $(\kappa_k(\boldsymbol{c}_1) + \kappa_k(\boldsymbol{c}_2) \cdot \kappa_k(\boldsymbol{s}))$, but in order to do that we would need another secret key, i.e., $\kappa_k(\boldsymbol{s})$. A straightforward approach is to generate multiple secret keys, each corresponding to a specific automorphism or rotation. One would then need

to track which automorphism or rotation was applied to a ciphertext to determine the correct secret key for decryption. More importantly, computing two ciphertexts derived from different rotations would be challenging. For these reasons, a more effective solution that uses key switching is preferred: namely after the application of an automorphism, the key is switched back from $\kappa_k(\boldsymbol{s})$ to $\boldsymbol{s}$. Note that an algorithm that requires many rotation will also require a large amount of memory as these keys are quite large, see e.g., Table 5 of [KPZ21].

Rotate$_{evk}(\boldsymbol{c}, i)$

Given a ciphertext $\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2$, return

$$\text{KeySwitch}_{evk}(\kappa_{5^i}(\boldsymbol{c}))$$

where KeySwitch perform a key switching from $\kappa_{5^i}(\boldsymbol{s})$ back to $\boldsymbol{s}$

Of course, for someone to be able to rotate the slots of a ciphertext, it is required that evk contains the corresponding key switching key (also called automorphism key, or Galois key).

3.3.1 Bootstrapping algorithms

The original CKKS scheme was extended one year after its presentation in [Che+18b], where the authors presented a way to bootstrap it. In the following years, many proposals improved this procedure, see e.g., [CCS19; Lee+21; BTH22; Kim+22a]. Among the latest works, we refer the reader to, e.g., [MLS25; Che+25; CKY25], which present the latest state-of-the-art algorithms to efficiently bootstrap a CKKS ciphertext.

The goal of this section is to give a broad view on how a general bootstrapping works, and in order to keep things "simple", we briefly recall how the original bootstrapping in [Che+18b] works, and we continue describing a (conceptually) simpler approach presented in [CCS19], although we refer the interested reader to the previously referenced papers. In "exact" schemes such as BGV, BFV, TFHE and so on, the goal of the bootstrapping is to reduce the amount of noise in the ciphertext to allow for further operations, using an encryption of the secret key and homomorphically evaluating the decryption circuit. On the other hand, in the case of CKKS, since the noise is part of the message, usually bootstrapping has the goal of resetting the available modulus that has been

[KPZ21]: A. Kim, Polyakov, and Zucca, *'Revisiting Homomorphic Encryption Schemes for Finite Fields'* . In: ASIACRYPT '21

[Che+18b]: Cheon, Han, A. Kim, M. Kim, et al., *'Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '18

[CCS19]: H. Chen, Chillotti, and Song, *'Improved Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '19

[Lee+21]: J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, et al., *'High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Polynomial Approximation and Inverse Sine Function'* . In: EUROCRYPT '21

[BTH22]: Bossuat, J. Troncoso-Pastoriza, and Hubaux, *'Bootstrapping for Approximate Homomorphic Encryption with Negligible Failure-Probability by Using Sparse-Secret Encapsulation'* . In: ACNS '22

[Kim+22a]: S. Kim, M. Park, J. Kim, T. Kim, et al., *'EvalRound Algorithm in CKKS Bootstrapping'* . In: ASIACRYPT '22

[MLS25]: Min, J.-W. Lee, and Song, *'Enhanced CKKS Bootstrapping with Generalized Polynomial Composites Approximation'* . In: ASIA CCS '25

[Che+25]: Cheon, Hanrot, J. Kim, and Stehlé (2025), SHIP: A Shallow and Highly Parallelizable CKKS Bootstrapping Algorithm (Cryptology ePrint Archive, Paper 2025/784)

[CKY25]: Cheon, J. Kim, and Yeo (2025), OverMod-Raise: Reducing Modulus Consumption of CKKS Bootstrapping (Cryptology ePrint Archive, Paper 2025/1298)

reduced during the evaluation of the homomorphic circuit due to the rescaling procedure. Moreover, it increases the error, as it is itself an approximate operation. So the ultimate goal of CKKS bootstrapping is to raise the current modulus q_ℓ back to its original level, so that further rescaling operations – hence, multiplications – can be performed.

3.3.1.1 Original CKKS bootstrapping

Let us assume that a ciphertext is at the last level q_1, and can be decrypted as $[\boldsymbol{c}_1 + \boldsymbol{c}_2 \cdot \boldsymbol{s}]_{q_1}$. An intuitive approach could be to simply raise the modulus of $\boldsymbol{c}$ back to say q_L, where q_L is the largest moduli of the ladder. Nevertheless, by decrypting the ciphertext, we would obtain something like

$$[\boldsymbol{c}_1 + \boldsymbol{c}_2 \cdot \boldsymbol{s}]_{q_L} \approx \boldsymbol{m} + q_1 \boldsymbol{t},$$

for some polynomial $\boldsymbol{t} \in \mathcal{R}_{q_L}$ such that $\|\boldsymbol{t}\|_\infty = O\sqrt{h}$, with h being the Hamming weight of the (binary) secret key $\boldsymbol{s}$. The goal of the bootstrapping operation is therefore to eliminate that $q_1 \boldsymbol{t}$ term. Theoretically, one approach could be to evaluate the modular reduction function $[\cdot]_{q_1}$ over the coefficients of the ciphertext, although this does not seem easy as the function must be approximated using polynomials (since the only arithmetic operations allowed are additions and multiplications).

In the first bootstrapping algorithm, presented in [Che+18b], the authors observed that if the message is not too large with respect to q_1, then the modular reduction can be approximated by a sine wave[2], as shown in Figure 62.

[Che+18b]: Cheon, Han, A. Kim, M. Kim, et al., *'Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '18

2: This is one of the reasons why, in practical implementations, the first modulus of the chain is larger than the others

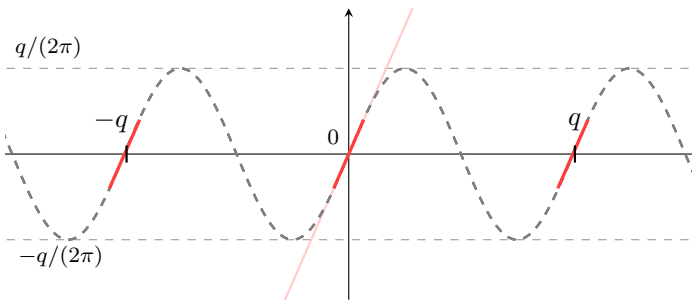


Figure 62: The modular reduction by q, in red, can be approximated by a sine function if the input is not too far from a multiple of q

By using the following formula:

$$S(t) := \frac{q_1}{2\pi} \sin\left(\frac{2\pi t}{q_1}\right), \tag{3.5}$$

we obtain a very good approximation of the modular reduction $[\cdot]_{q_1}$, when the input value $\boldsymbol{m} + q_1\boldsymbol{t}$ is close to a multiple of q_1, namely when $\boldsymbol{t}, \boldsymbol{m}$ are not too large (with respect to q_1) and q_1 dominates the expression. In other words, this method works as long as $\boldsymbol{m}$ is short, given that the secret $\boldsymbol{s}$ is short. A brief analysis shows that this error is indeed small when $m < q^{2/3}$:

$$\begin{aligned} |[t]_q - S(t)| &= \frac{q}{2\pi}\left|\frac{2\pi m}{q} - \sin\left(\frac{2\pi m}{q}\right)\right| \\ &\leq \frac{q}{2\pi}\cdot\frac{1}{3!}\left(\frac{2\pi|m|}{q}\right)^3 = O\left(q\cdot\left(\frac{|m|}{q}\right)^3\right) \end{aligned}$$

However, just like the modular reduction, the sine function is not polynomial and can not be evaluated natively in CKKS. We remark that this is not the best approach to approximate such a function. Indeed, many more interesting way of approximating the modular reduction were given in the subsequent years.

3.3.1.2 An efficient approach using Chebyshev polynomials

Of particular interest, there is a proposal by Chen, Chillotti and Song [CCS19], where they took advantage of Chebyshev polynomials to approximate the sine wave. We will therefore present this approach, but we refer to the ones cited at the beginning of Section 3.3.1 for more modern (and slightly more complex) proposals of bootstrapping.

[CCS19]: H. Chen, Chillotti, and Song, *'Improved Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '19

Modulus raise. The very first procedure of the bootstrapping is, given a ciphertext say at level 1 to raise its modulus from q_1 back to q_L. So simply change the modulus of $\boldsymbol{c} \in \mathcal{R}^2_{q_1}$ to $\boldsymbol{c} \in \mathcal{R}^2_{q_L}$. However, at this point, if we try to decrypt the ciphertext using the secret, we will obtain

$$\boldsymbol{c}(\boldsymbol{s}) \approx \boldsymbol{m} + q_1 \cdot \boldsymbol{t} \in \mathcal{R}_{q_L}$$

with $\boldsymbol{t}$ being some polynomial. Notice that, since the secret $\boldsymbol{s}$ has small coefficients, so it will $\boldsymbol{t}$, this is also what was noticed in [Che+18b]. The rest of the procedures consist in removing the $q_1 \cdot \boldsymbol{t}$ term from the decryption of $\boldsymbol{c}$. A remaining challenge is that homomorphic operations in CKKS act on plaintext slots rather than on polynomial coefficients, but the term we want to remove lies in the coefficients. Therefore, before and after computing the approximation of the modular reduction, we need to convert the coefficients into slots and then back

[Che+18b]: Cheon, Han, A. Kim, M. Kim, et al., *'Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '18

again. This conversion can be done by homomorphically evaluating the encoding and decoding steps, which are linear transformations on plaintext vectors.

Coefficients → slots. The next procedure consists in taking the coefficients of $\boldsymbol{c}$ and putting them in *Slots* representation using the encoding/IDFT procedure. In particular, given the following matrix:

$$F = \begin{pmatrix} 1 & \zeta & \zeta^2 & \dots & \zeta^{n/2-1} \\ 1 & \zeta^5 & \zeta^{2\cdot 5} & \dots & \zeta^{5(n/2-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{2n-3} & \zeta^{2(2n-3)} & \dots & \zeta^{(2n-3)(n/2-1)} \end{pmatrix}, \quad (3.6)$$

where $\zeta = \xi_{2n}$. The `CoeffsToSlots` procedure simply computes the following transformation:

$$\boldsymbol{p} \mapsto F \cdot \boldsymbol{p},$$

where $\boldsymbol{p}$ is the plaintext encoded inside some ciphertext $\boldsymbol{c}$. This homomorphic linear transformation can be evaluated in many different ways, and the aspects to consider are many: we want to minimize runtime, noise growth, levels consumption, memory requirements, and so on. In [CCS19] they suggest to use FFT-like algorithms to speed up the process [CCS19, Algorithm 1]. In particular, they propose algorithms that require slightly more than 1 multiplication (i.e., $\log(n/2)$) and $\log(n/2)$ operations. Additionally, they propose, building upon [HS14], some "level collapsing" methods that allow to collapse (or merge) two subsequent linear transformations into one. As an example, think of a sequence of ℓ linear transformations $T_1 \circ T_2 \circ \dots \circ T_\ell$. One could evaluate them all sequentially consuming ℓ levels. Or, one could only use $\log(\ell)$ levels by collapsing some transformations into one, e.g., as:

[HS14]: Halevi and Shoup, *'Algorithms in HElib'* . In: CRYPTO '14

$$M_1 = (T_1 \circ T_2), \quad M_2 = (T_3 \circ T_4),$$

and then applying $M_1 \circ M_2$ only, consuming two levels instead of four. In particular, the approach adopted by the authors has been to associate a cost function to each homomorphic operation (i.e., ciphertext/ciphertext addition, ciphertext/-ciphertext multiplication, ciphertext/plaintext addition, ciphertext/plaintext multiplication, rotation), and the goal has been to minimize such cost function. Section 3.3 of [CCS19] present some optimal strategies, with the goal of minimizing the complexity of the algorithm at the cost of increasing the number of required levels. Of course, one might perform the

whole operation using a single level, but this has enormous complexity and will not pay off in the long term.

Evaluating the sine wave. Remark that, when the coefficients to slots transformation has been evaluated, we have to remove the $q_1 \cdot \boldsymbol{t}$ term by evaluating an approximation of $[\cdot]_{q_1}$. We can reformulate (3.5), assuming $t \in [-Kq, Kq]$ for some small $K \in \mathbb{Z}^+$ as:

$$g(x) := \frac{1}{2\pi} \sin(2\pi K x) \tag{3.7}$$

for $x \in [-1, 1]$. It turns out that a nice approach to find a (quasi) optimal approximation is to use Chebyshev polynomials.

Chebyshev polynomials. The Chebyshev polynomials [Tre19] refer to a family of orthogonal polynomials defined by the following recurrence relation:

[Tre19]: Trefethen (2019), Approximation Theory and Approximation Practice, Extended Edition

$$\begin{aligned} T_0(x) &= 1, \quad T_1(x) = x \\ T_{2n}(x) &= 2T_n(x)^2 - 1 \\ T_{2n+1}(x) &= 2T_n(x) \cdot T_{n+1}(x) - x. \end{aligned}$$

Given this set of polynomials, it is also possible to define the so-called *Chebyshev series* representation of a polynomial:

$$p_n(x) = \sum_{k=0}^{n-1} c_k T_k(x),$$

where $p_n(x)$ is a polynomial of degree $n-1$, $T_k(x)$ is the k-th Chebyshev polynomial of the first kind, and each coefficient c_k is set according to a certain function f to be approximated, without loss of generality, over $[-1, 1]$. Given the following set, which we will call *Chebyshev interpolants*:

$$x_i = \cos\left(\frac{(i + \frac{1}{2})\pi}{n}\right) \text{ with } 0 \leq i < n,$$

the coefficients c_k are determined such that $p_n(x_i) = f(x_i)$. Putting everything together, it is possible to define polynomial approximations of any continuous function f by performing a polynomial regression over the Chebyshev interpolants. In particular, an approximation of degree d is performed over the $d+1$ roots.

Theorem 9 (*Chebyshev roots are "near-best"* [EZ66])

Let f be a continuous function on $[-1, 1]$, let p denote its interpolant in n Chebyshev points and let p^* be its best degree n approximation with respect to the infinity norm. It holds that:

$$||f - p||_\infty \leq ||f - p^*||_\infty \cdot \left(2 + \frac{2}{\pi}\log(n)\right).$$

Informally, Theorem 9 ensures that, by using Chebyshev roots, the approximation is not too far from the best.

Hence, one strategy to evaluate $g(x)$ is via Chebyshev polynomials. In [CCS19] we can find an adapted version of the Paterson-Stockmeyer (PS) algorithm [PS73] to evaluate these polynomials requiring only $O(\sqrt{n})$ levels of a degree n polynomial. The main issue of PS is that it assumes as input a polynomial in power base $1, X, \ldots, X^n$. One could transform the polynomial obtained through a Chebyshev interpolation in power basis using e.g., a transition matrix. Nevertheless, such matrix is very ill-conditioned (its conditional number grows exponentially with n [Bec97]), so the strategy proposed in [CCS19] is to adapt PS to be run over Chebyshev basis. A description of the algorithm can be found in the referenced paper.

Slots → coefficients. As a last step of the bootstrapping, we have the inverse transformation `SlotsToCoeffs`, which takes the slots obtained after the modulus raise phase and takes them back to coefficients representation by simply evaluating

$$\boldsymbol{p} \mapsto F^{-1} \cdot \boldsymbol{p},$$

where F is the matrix described in (3.6). As a result, we obtain a ciphertext that is now correctly decryptable (although with larger error than the original one) but at a larger level $q_i > q_1$, with i being slightly smaller than L due to the two linear transformations and the evaluation of the (approximate) modular reduction.

To conclude, we believe that the method described in [CCS19] is conceptually simple and well explained in the paper. Nonetheless, today we have more efficient algorithms to perform CKKS bootstrapping, we give some references for the interested reader, e.g., [Kim+22b; BTH22; MLS25; Che+25; CKY25; MLS25].

Moreover, we also refer to one of the first practical examples of *functional bootstrapping* in CKKS [AKP25], namely a boot-

[PS73]: Paterson and Stockmeyer (1973), *'On the number of nonscalar multiplications necessary to evaluate polynomials'* . In: SIAM Journal on Computing

[Bec97]: Beckermann (1997), *'On the numerical condition of polynomial bases: estimates for the condition number of Vandermonde, Krylov and Hankel matrices'*

[Kim+22b]: S. Kim, M. Park, J. Kim, T. Kim, et al., *'EvalRound algorithm in CKKS bootstrapping'* . In: ASIACRYPT '22

[BTH22]: Bossuat, J. Troncoso-Pastoriza, and Hubaux, *'Bootstrapping for Approximate Homomorphic Encryption with Negligible Failure-Probability by Using Sparse-Secret Encapsulation'* . In: ACNS '22

[MLS25]: Min, J.-W. Lee, and Song, *'Enhanced CKKS Bootstrapping with Generalized Polynomial Composites Approximation'* . In: ASIA CCS '25

[Che+25]: Cheon, Hanrot, J. Kim, and Stehlé (2025), SHIP: A Shallow and Highly Parallelizable CKKS Bootstrapping Algorithm (Cryptology ePrint Archive, Paper 2025/784)

[CKY25]: Cheon, J. Kim, and Yeo (2025), OverModRaise: Reducing Modulus Consumption of CKKS Bootstrapping (Cryptology ePrint Archive, Paper 2025/1298)

[MLS25]: Min, J.-W. Lee, and Song, *'Enhanced CKKS Bootstrapping with Generalized Polynomial Composites Approximation'* . In: ASIA CCS '25

[AKP25]: Alexandru, A. Kim, and Polyakov, *'General Functional Bootstrapping Using CKKS'* . In: CRYPTO '25

strapping that, along with the modular reduction, allows one to evaluate an arbitrary look-up table.

The CKKS scheme will be the main building block for the algorithms presented in Chapters 5, 6 and 7. Before that, we take a step back and we focus on "simpler" lattice-based FHE cryptosystems, and how to build them taking a fully lattice and geometrical perspective.

Beyond LWE: a general lattice framework

4

This chapter is an extended/revised version of 'Beyond LWE: a Lattice Framework for Homomorphic Encryption' *co-authored with A. Leporati and W. van Woerden, currently under review at PKC '26*

Abstract: We propose a generalization of homomorphic encryption (HE) schemes from a purely geometrical and lattice-based perspective. Most of the current reference HE schemes are based on the Learning with Errors (LWE) problem or structured versions thereof. In this proposal, we first investigate LWE-based cryptosystems from a lattice point of view and present a framework that allows to obtain the same result, in geometrical terms, from any lattice – as long as it contains a sufficiently short trapdoor vector. More precisely, we generalize the classical BGV (Brakerski, Gentry and Vaikuntanathan, ITCS '12) and GSW (Gentry, Sahai and Waters, CRYPTO '14) schemes to purely lattice-based variants, which we call Lattice-BGV and Lattice-GSW. By abstracting away the particular hardness assumption, our lattice framework allows to be instantiated with a broader range of lattice families and hardness assumptions. For example, LWE gives a natural trapdoor for random q-ary lattices, and when plugged into our framework one obtains the original BGV and GSW schemes, while in this work we will also consider an instantiation based on the Lattice Isomorphism Problem (LIP), leading to the first more advanced cryptographic scheme build from LIP [1]. Our framework also gives a geometrical and natural explanation of HE procedures and generalizes some properties, such as message packing, the ability to store many messages in a single ciphertext, without relying on any particular algebraic structure.

1: In a concurrent work Branco, Malavolta and Maradni [BMM25] propose an alternative LIP-based FHE construction.

4.1 Introduction

RIGINALLY referred to as *privacy homomorphisms*, the theoretical concept of homomorphic encryption was introduced in 1978 by Rivest, Adleman and Dertouzos [RAD78]. A homomorphic encryption scheme allows for computations to be performed on encrypted data without

[RAD78]: R. L. Rivest, Adleman, and Dertouzos (1978), *'On data banks and privacy homomorphisms'* . In: Foundations of secure computation

having access to the secret key. Since this theoretical formulation, several *partially* HE schemes, i.e., limited to the evaluation of one type of operation only (either addition or multiplication), were introduced [GM82; ElG85; Pai99; BGN05].

Both these approaches are based on the same hardness assumption, that is the Learning with Errors (LWE) problem [Reg05] or structured versions thereof. Indeed, most of the more popular FHE schemes share the same key generation procedure [Bou+20] This begs the question: is LWE required to instantiate a HE scheme, or can we also use other lattice assumptions? Or related, what are the minimum requirements to instantiate it? For the sake of clarity, we will make no difference between HE and FHE as, for the appropriate parameters, our HE schemes can be naturally extended to FHE schemes by a bootstrapping procedure.

[GM82]: Goldwasser and Micali, *'Probabilistic encryption & how to play mental poker keeping secret all partial information'* . In: STOC '82

[ElG85]: ElGamal, *'A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms'* . In: CRYPTO '84

[Pai99]: Paillier, *'Public-Key Cryptosystems Based on Composite Degree Residuosity Classes'* . In: EUROCRYPT '99

[BGN05]: Boneh, Goh, and Nissim, *'Evaluating 2-DNF Formulas on Ciphertexts'* . In: TCC '05

[Reg05]: Regev, *'On lattices, learning with errors, random linear codes, and cryptography'* . In: STOC '05

A lattice framework for (fully) homomorphic encryption. To answer these questions we go back to the core of these LWE-based HE schemes to better understand their workings. Since LWE can be seen as a lattice problem, we will take a lattice perspective to describe them. This allows to have some geometrical intuition on both the key generation, the encryption/decryption, and the other FHE specific procedures. In particular, it makes clear what *lattice properties and algorithms* these schemes are actually using, and what the requirements are for the lattices involved.

Following this perspective, we see that the LWE assumption is used in two ways for the underlying encryption scheme. Firstly, in the key generation, it is used to create a random q-ary lattice $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$ with a planted secret short vector $\boldsymbol{s} \in \mathcal{L}$. Secondly, for encryption, a message is encoded in a small error $\boldsymbol{e} \in \mathbb{Z}^m$ which is then hidden as a Bounded Distance Decoding (BDD) target $\boldsymbol{t} \in \boldsymbol{e} + q\mathcal{L}^*$ in the dual lattice $\mathcal{L}^*$ scaled up by q. By the LWE assumption, this BDD target is indistinguishable from a uniform target and thus the encoded message is securely hidden. To decrypt, one then uses the secret short vector $\boldsymbol{s} \in \mathcal{L}$ to partially decode the target, as $\langle \boldsymbol{t}, \boldsymbol{s} \rangle \equiv \langle \boldsymbol{e}, \boldsymbol{s} \rangle \bmod q$. Since $\langle \boldsymbol{e}, \boldsymbol{s} \rangle \in \mathbb{Z}$ is small, this value is recovered exactly, and we obtain information about the error $\boldsymbol{e}$ which, if well encoded, allows to recover the message.

It is folklore that the above strategy can also be generalized to construct a simple encryption scheme using different lattice families with different underlying security assumptions. It turns out however that these minimal requirements are also sufficient for HE. To support all common HE primitives, one

only requires the key generation to produce a *public lattice* $\mathcal{L}$, for which decoding in the dual lattice $\mathcal{L}^*$ is hard, along with a *secret short trapdoor vector* $\boldsymbol{s} \in \mathcal{L}$. We demonstrate this by presenting a *lattice framework* for HE. In particular, we present an explicit generalization of the classical BGV and GSW schemes using only this public lattice and the secret short trapdoor vector. This is a proper generalization: if instantiated with an LWE-based key generation, we obtain the original schemes.

A high-level view on the homomorphic operations. Recall that ciphertexts can be seen as BDD targets with respect to the lattice $q\mathcal{L}^*$. An important step in this generalization is the discretized ambient space $\mathcal{D} \supset q\mathcal{L}^*$ in which these BDD targets live. We show that it is sufficient that $\mathcal{L}^* \subset \mathcal{D} \subset q\mathcal{L}^*$ for some modulus q and such that we have a relatively good public basis D of $\mathcal{D}$. Note that these properties generalize the q-ary integer lattices and their orthogonal ambient space $\mathbb{Z}^m$. The former creates a ciphertext space $\mathcal{C} := \mathcal{D}/q\mathcal{L}^* \cong G \subset \mathbb{Z}^m/q\mathbb{Z}^m$ whose group structure is required to make the homomorphic operations succeed, while the good basis D is required to sample short errors $\boldsymbol{e} \in \mathcal{D}$ (discrete Gaussian sampling) and for modulus switching (decoding to the ambient space).

One way to achieve this is to pick $\mathcal{D} := \mathcal{L}^*$ with D equal to its public basis and the modulus q sufficiently large (to make the basis D relatively good). We also demonstrate a more efficient construction, which further compresses the ambient space, which could be interpreted as a generalization of small-secret LWE.

With the ambient and ciphertext space settled we show that also in this more general setting, multiplications can naturally be performed by a tensor product $\boldsymbol{c}_1 \otimes \boldsymbol{c}_2 \in \mathcal{C} \otimes \mathcal{C}$ between ciphertexts or equivalently their errors $\boldsymbol{e}_1 \otimes \boldsymbol{e}_2$, just as for BGV. Given the right encoding of the message space $\mathcal{M}$ into errors this leads to a tensor product of the messages $\boldsymbol{m}_1 \otimes \boldsymbol{m}_2 \in \mathcal{M} \otimes \mathcal{M}$ (which, e.g., for the special case $\mathcal{M} \cong \mathbb{Z}/2\mathbb{Z}$ is a binary multiplication $\mu_1 \cdot \mu_2$). On the lattice side this means we move from a secret trapdoor vector $\boldsymbol{s} \in \mathcal{L}$ and the q-dual $q\mathcal{L}^*$ to the trapdoor vector $\boldsymbol{s} \otimes \boldsymbol{s} \in \mathcal{L} \times \mathcal{L}$ and the q-dual $q(\mathcal{L}^* \otimes \mathcal{L}^*)$.

We show how the key switching procedure allows us to move from this large-dimensional tensored space back to any new keypair $\boldsymbol{s}', \mathcal{L}'$ and ambient space $\mathcal{D}'$ using the same modulus,

along with maintaining (part of) the message $\boldsymbol{m}_1 \otimes \boldsymbol{m}_2 \in \mathcal{M} \otimes \mathcal{M}$.

To decrease the absolute decoding errors we also demonstrate a general modulus switching technique from one ambient space $q_1\mathcal{L}^* \subset \mathcal{D}_1 \subset \mathcal{L}^*$ to a relatively sparser ambient space $q_1\mathcal{L}^* \subset (q_1/q_2)\mathcal{D}_2 \subset (q_1/q_2)\mathcal{L}^*$ where $q_2 \mid q_1$. We show that this procedure is precisely an (approximate) decoding problem from $\mathcal{D}_1$ to the sparser ambient space $(q_1/q_2)\mathcal{D}_2$, which we can perform using its good basis D_2. Some extra care has to be taken depending on the message encoding. After scaling down by q_1/q_2 the new error becomes $(\boldsymbol{e} + \boldsymbol{\delta})/(q_1/q_2)$ where $\boldsymbol{e}$ was the old ciphertext error, and $\boldsymbol{\delta}$ the ambient space rounding error. This also shows that an ambient space with better decoding properties could lead to better modulus switching.

Together these operations form the basis of our Lattice-BGV framework. We also consider the GSW approach which uses a redundant encoding of the messages into multiple BGV ciphertexts at the benefit of a natural matrix-multiplication that avoids the large increase of dimension, leading to the Lattice-GSW framework.

Lattice isomorphism problem. Our lattice framework can directly be instantiated with lattices constructed from LWE, NTRU and related assumptions. However, a new assumption of interest that fits well within our framework is the Lattice isomorphism problem (Section 2.5): Two lattices $\mathcal{L}_1, \mathcal{L}_2 \subset \mathbb{R}^n$ are called *isomorphic* if there exists a linear isometry mapping one to the other, i.e., if there exists an orthonormal transformation $O \in \mathcal{O}_m(\mathbb{R})$ such that $\mathcal{L}_2 = O \cdot \mathcal{L}_1 = \{O\boldsymbol{v} : \boldsymbol{v} \in \mathcal{L}_1\}$. LIP then asks, given two isomorphic lattices, to compute such an orthonormal transformation mapping one lattice to the other. The main idea is that one can use such an an isometry to hide the structure of a lattice.

Since the usage of LIP as a hardness assumption is relatively new, there are at this moment only LIP-based schemes for basic cryptographic primitives, like identification, encryption and signature schemes [DvW22; Ben+23b; Duc+22]. Given our lattice framework, we can however also construct a HE scheme based on LIP in a straightforward way. For this we can pick any lattice $\mathcal{L}'$ with some known sufficiently short vector $\boldsymbol{s}' \in \mathcal{L}'$[2].

Then we hide the known structure by considering a bad public basis of a lattice $\mathcal{L} := O \cdot \mathcal{L}'$ isomorphic to $\mathcal{L}'$ by some *secret*

[DvW22]: Ducas and van Woerden, *'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'* . In: EUROCRYPT '22

[Ben+23b]: Bennett, Ganju, Peetathawatchai, and Stephens-Davidowitz, *'Just how hard are rotations of $\mathbb{Z}^n$? Algorithms and cryptography with the simplest lattice'* . In: EUROCRYPT '23

[Duc+22]: Ducas, Postlethwaite, Pulles, and W. v. Woerden, *'Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple'* . In: ASIACRYPT '22

2: For example, as the maybe simplest case, one could consider $\mathcal{L}' = \mathbb{Z}^m$ and $\boldsymbol{s}' \in \mathbb{Z}^m$ one of the unit vectors

isometry $O \in \mathcal{O}_m(\mathbb{R})$. Given the secret isometry O one also knows a secret short vector $\boldsymbol{s} = O \cdot \boldsymbol{s}' \in \mathcal{L}$. Informally, with only the bad description of $\mathcal{L}$, and being hard to recover the isometry, it is typically hard to decode a BDD instance in $q\mathcal{L}^*$ and therefore the scheme is secure.

This can be formally reduced to a ΔLIP (Definition 53) assumption as in [DvW22], although we also present a more natural IND-CPA security proof based on two assumptions inspired by results from [vW24].

[vW24]: van Woerden, *'Dense and smooth lattices in any genus'* . In: ASIACRYPT '24

We thus have all the ingredients to instantiate our lattice framework based on the LIP assumption.

Connections with [BMM25]. In a recent concurrent work, Branco, Malavolta and Maradni [BMM25] propose an FHE scheme based on the hardness of LIP. While this does not coincide with the main goal of our work, we do want to shortly compare it to a LIP-based instantiation of our lattice framework. We will mostly focus on the main differences.

[BMM25]: Branco, Malavolta, and Maradni (2025), Fully-Homomorphic Encryption from Lattice Isomorphism (Cryptology ePrint Archive, Paper 2025/993)

Firstly, the work of [BMM25] has a large focus on building an FHE scheme along with a security reduction to (distinguish-)LIP with a similar strategy as used in [DvW22]. As our main goal is to give a general lattice framework for (F)HE and its operations, we left such a security proof out of the scope of our work. We do believe the reduction of [DvW22] could also be adapted to our work.

[DvW22]: Ducas and van Woerden, *'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'* . In: EUROCRYPT '22

Secondly, [BMM25] presents a private-key encryption scheme, contrary to our public-key encryption scheme. This is vital for the correctness of their homomorphic multiplications. Following the strategy of [Rot11] one could turn their private-key FHE scheme into a public-key. Conversely, both our GSW and BGV-like multiplication approaches naturally work in a public-key setting, without resorting to extra constructions à la [Rot11].

[Rot11]: Rothblum, *'Homomorphic Encryption: From Private-Key to Public-Key'* . In: TCC '11

Thirdly, the schemes we present naturally allow for the evaluation of circuits of depth more than one and we give an explicit connection between the shortness of the secret trapdoor vector and the depth that can be achieved before bootstrapping is required. In [BMM25] only branching programs are considered, where one of the two inputs of the homomorphic multiplication is a fresh ciphertext.

Lastly, while [BMM25] claims an *FHE* scheme our work claims an *HE* scheme. Concretely however, both works only present an HE scheme for bounded depth circuits, and argue that

from there bootstrapping is possible by standard techniques. To conclude, we mostly consider the goals and execution of our work and [BMM25] as orthogonal.

Structure of the chapter. The rest of the chapter is structured as follows. Section 4.2 will contain all the required preliminaries from lattices and the BGV/GSW definitions. Section 4.3 will introduce the lattice perspective on the fundamental components of the previous schemes, namely the key generation and encryption/decryption procedures. Section 4.5 will introduce a pure lattice-based version of both BGV and GSW. Section 4.6 will present a possible LIP-based instantiation of previously introduced schemes. Eventually, in Section 4.7, we consider some further extensions and open questions.

4.2 Preliminaries

In what follows, if not stated otherwise, lowercase letters are used to indicate numbers (e.g. a), bold-italic lowercase letters to indicate vectors (e.g. $\boldsymbol{v}$), and uppercase letters are used to indicate matrices (e.g. A). Vectors are considered as column vectors, the i-th column of a matrix A is referred to as $\boldsymbol{a}_i$ and the i-th element of a vector $\boldsymbol{v}$ as v_i. The symbol $\mathbb{Z}_q$ refers to the cyclic group $\mathbb{Z}/q\mathbb{Z}$ of integers modulo q, which we identify with $[-q/2, \dots, q/2) \cap \mathbb{Z}$.

The general linear group of invertible matrices with integer (or real) entries of dimension m is referred to as $\mathrm{GL}_m(\mathbb{Z})$ (or $\mathrm{GL}_m(\mathbb{R})$, respectively). The group of orthonormal matrices of dimension m is referred to as $\mathcal{O}_m(\mathbb{R})$. The transpose of a matrix A or vector $\boldsymbol{v}$ is referred to as A^t, and to its inverse as A^{-t}. Given a matrix B composed of vectors $\boldsymbol{b}_i$, we write $\|B\| := \max_i \|\boldsymbol{b}_i\|$, and will refer to its Gram-Schmidt orthogonalization as $\tilde{B}$ and to the Gram-Schmidt vectors as $\tilde{\boldsymbol{b}}_i$.

The notation $a \sim \chi$ indicates a value a sampled from the probability distribution χ. The notation $[a]_q : \mathbb{Z} \to \mathbb{Z}_q$ indicates the modulo reduction a mod q in the interval $[-q/2, q/2)$. The notation $\lfloor a \rceil_p : \mathbb{Z}_q \to \{0, \dots, p-1\}$, indicates a function that returns the nearest integer to $a \cdot p/q$, namely $\lfloor a \rceil_p := \lfloor a \cdot p/q + 1/2 \rfloor \bmod p$.

We write $\log := \log_2$ for the logarithm function with base two, i.e., $\log(2^k) = k$. Let $\mathtt{Bits}(a) : \mathbb{Z}_q \to \{0,1\}^{\log(q)} \subset \mathbb{Z}_q^{\log(q)}$ be the bit decomposition of a, least to most significant bit,

and let $\texttt{Pow}(a) : \mathbb{Z}_q \to \mathbb{Z}_q^{\log(q)} = (a, 2a, 4a, \ldots, 2^{\log(q)-1} \cdot a)$. By abuse of notation we allow both functions to be applied to vectors by evaluating the corresponding function independently on each entry, giving functions $\texttt{Bit}, \texttt{Pow} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \log(q)}$.

We will use the LWE definition stated in Definition 32. We emphasize and stress that an LWE instance is defined using the notation introduced there, namely $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{e} \bmod q$, where the "error" $\boldsymbol{s} \in \mathbb{Z}_q^m$ is a small vector from the distribution χ. Although not standard, this notation will allow to have a similar notation between the original LWE-based schemes and our generalizations, making later arguments more readable. We will later see that recovering the $(\boldsymbol{x}, \boldsymbol{s})$ pair from an LWE instance is essentially a syndrome decoding problem in a random q-ary lattice.

4.2.1 Public key encryption scheme

While we consider homomorphic encryption (HE) schemes in this work, we will only consider the underlying plain public key encryption (PKE) schemes for their security.

Definition 61 (*PKE Scheme*)

A Public Key Encryption *scheme* Π *consists of a message space* $\mathcal{M}$*, a public key space* $\mathcal{K}_p$*, a secret key space* $\mathcal{K}_s$*, a ciphertext space* $\mathcal{C}$*, and three probabilistic polynomial-time (PPT) algorithms* (gen, enc, dec) *such that* $gen : \{0,1\}^\lambda \to \mathcal{K}_p \times \mathcal{K}_s$ *(key generation),* $enc : \mathcal{K}_p \times \mathcal{M} \to \mathcal{C}$ *(encryption) and* $dec : \mathcal{K}_s \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$*, such that for any* $(pk, sk) \leftarrow gen(1^\lambda)$ *and* $m \in \mathcal{M}$ *it holds that* $dec_{sk}(enc_{pk}(m)) = m$ *with overwhelming probability in* λ *(correctness).*

For an HE scheme one has to add a (limited) evaluation function that can act on ciphertext with an optional evaluation key. In this work the evaluation key will have the form of encryptions of (an encoding of) the secret key. For simplicity we will assume that these encryptions are performed under a fresh keypair, so as not have to consider any circular security assumptions. As a result we can also simply consider the IND-CPA security of the underlying encryption scheme (Definition 55).

We observe that, since Li and Micciancio attack [LM21], it is required for HE schemes to also answer to a more advanced security model, that is IND-CPA$^{\text{D}}$. While discussing such definition is out of scope for this work – although we refer to, e.g.,

[LM21]: Li and Micciancio, *'On the Security of Homomorphic Encryption on Approximate Numbers'* . In: EUROCRYPT '21

[Che+24b] – we highly believe that existing countermeasures for LWE-based schemes can easily be ported in our framework as well. For simplicity, we can assume that either the schemes are instantiated with sets of parameters such that decryption errors happen with negligible probability, or that decryption results are not shared with any possible attacker.

[Che+24b]: Cheon, Choe, Passelègue, Stehlé, et al., *'Attacks Against the IND-CPA^D Security of Exact FHE Schemes'* . In: CCS '24

4.2.2 Original LWE-based BGV cryptosystem

Started by the Brakerski-Gentry-Vaikuntanathan (BGV) cryptosystem [BGV12], the BGV-like type of schemes allows for practical homomorphic circuit evaluation based on (Ring-)LWE. The adopted "leveled" approach allows to evaluate circuits of a fixed multiplicative depth – or levels – depending on the selected set of parameters.

[BGV12]: Brakerski, Gentry, and Vaikuntanathan, *'(Leveled) fully homomorphic encryption without bootstrapping'* . In: ITCS '12

Differently from Section 3.2, we describe the public-key version of the BGV cryptosystem in $\mathrm{LWE}_{n-1,m,q,\chi}$ form. Although the ring version based on RLWE is more efficient, we focus on the plain variant to simplify the geometric interpretation we will give later – although similar arguments can be applied by considering RLWE as a particular case of LWE. Let $n, q \in \mathbb{N}$ be the LWE dimension and modulus respectively, and let $m = \mathcal{O}(n \log(q))$ be the number of LWE samples. We also define $t \ll q$ with $\gcd(t, q) = 1$ as the plaintext modulus (i.e., the message space $\mathcal{M} = \mathbb{Z}_t$) and χ as a small distribution from $\mathbb{Z}_q$. The ciphertext space $\mathcal{C}$ is identified with $\mathbb{Z}_q^n$. The starting modulus q is constructed as $q = \prod_{i=1}^{L+1} p_i$, where L is the multiplicative depth of the circuit to be evaluated and each p_i is set to be $p_i \equiv 1 \bmod t$. More advanced instantiations, such as the Gentry-Halevi-Smart [GHS12b] one, avoid constraints on the modulus by tweaking the encryption algorithm, although we stick for simplicity to the original formulation.

[GHS12b]: Gentry, Halevi, and Smart, *'Homomorphic Evaluation of the AES Circuit'* . In: CRYPTO '12

- Gen(1^λ): pick $n = n(\lambda), m = \mathcal{O}(n \log(q))$, and let the secret $sk = \boldsymbol{x} := (1, -\boldsymbol{x'}) \in \mathbb{Z}_q^n$, where $\boldsymbol{x'} \sim \mathbb{Z}_q^{n-1}$ is uniformly sampled. The public key is obtained by sampling a uniformly random $B \sim \mathbb{Z}_q^{m \times (n-1)}$ and by defining the LWE instance $\boldsymbol{b} := B\boldsymbol{x'} + \boldsymbol{s} \in \mathbb{Z}_q^m$, for some small $\boldsymbol{s} \sim \chi^m$. Then, generate the public key A by appending $\boldsymbol{b}$ as a column and B as:

$$\begin{aligned} sk &= \boldsymbol{x} := (1, -\boldsymbol{x'}) \in \mathbb{Z}_q^n, \\ pk &= A := \left[\, \boldsymbol{b} \mid B \,\right] \in \mathbb{Z}_q^{m \times n}. \end{aligned} \tag{4.1}$$

  Note that $A \cdot \boldsymbol{x} = \boldsymbol{s} \in \mathbb{Z}_q^m$.

- $\mathrm{Enc}_{pk}(\mu)$: the encryption function under a fixed public key pk is defined as:

$$\begin{aligned} &\mathrm{BGV.Enc}_{pk} : \mathcal{M} \to \mathcal{C}, \\ &\mathrm{BGV.Enc}_{pk}(\mu) := [t \cdot A^t \cdot \boldsymbol{r} + \boldsymbol{m}]_q \in \mathcal{C}, \end{aligned} \tag{4.2}$$

  where $\mu \in \mathcal{M}$, $\boldsymbol{m} := (\mu, 0, \ldots, 0)$ and $\boldsymbol{r} \sim \{0,1\}^m$ is uniformly sampled.
- $\mathrm{Dec}_{sk}(\boldsymbol{c})$: the decryption function under a fixed secret key sk is defined as

$$\begin{aligned} &\mathrm{BGV.Dec}_{sk} : \mathcal{C} \to \mathcal{M}, \\ &\mathrm{BGV.Dec}_{sk}(\boldsymbol{c}) := [[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q]_t \in \mathcal{M}, \end{aligned} \tag{4.3}$$

  correctness can be shown by observing that $pk \cdot sk = t\boldsymbol{s}$ is short.
- $\mathrm{Add}(\boldsymbol{c}, \boldsymbol{c}')$: to homomorphically add two underlying messages, evaluate an addition between their corresponding ciphertexts:

$$\begin{aligned} &\mathrm{BGV.Add} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}, \\ &\mathrm{BGV.Add}(\boldsymbol{c}, \boldsymbol{c}') := \boldsymbol{c} + \boldsymbol{c}' \in \mathcal{C}. \end{aligned} \tag{4.4}$$

- $\mathrm{Mul}(\boldsymbol{c}, \boldsymbol{c}')$: to homomorphically multiply two underlying messages, evaluate a tensor product between their corresponding ciphertexts:

$$\begin{aligned} &\mathrm{BGV.Mul} : \mathcal{C} \times \mathcal{C} \to \mathcal{C} \otimes \mathcal{C}, \\ &\mathrm{BGV.Mul}(\boldsymbol{c}, \boldsymbol{c}') := \boldsymbol{c} \otimes \boldsymbol{c}' \in \mathcal{C} \otimes \mathcal{C}. \end{aligned} \tag{4.5}$$

We can prove the security of BGV by (i) replacing the LWE sample $\boldsymbol{b}$ with a uniformly random vector and (ii) using the leftover hash lemma to argue that the ciphertext is indistinguishable from the uniform distribution over $\mathbb{Z}_q^n$.

Modulus switching. We call $[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q$ the *decryption error* for BGV. Assuming the initial decryption error in a fresh ciphertext to be bounded by b, the output of an addition will be bounded by $2b$ and the multiplication by roughly b^2. By choosing $q = \Theta(b^{L+1})$, one would be allowed to approximately perform up to $\log(L)$ multiplications since the noise grows exponentially. A possible solution, firstly suggested in [BV11], is to perform a *modulus switching*. The informal idea is that the error growth is mostly determined by the absolute size of the errors, and one can reduce the absolute value of the error in the ciphertext by "cutting" part of the current modulus after

each multiplication. Although the relative error with respect to the modulus remains fixed, the absolute error is reduced and this enables to perform more computations with a smaller error growth.

Say that a fresh ciphertext is encrypted using modulus q_1 such that $q_1 \equiv 1 \bmod t$, and let q_2 be the new modulus such that $q_2 \mid q_1$ and $q_2 \ll q_1$. Given a ciphertext $\boldsymbol{c} \in \mathbb{Z}_{q_1}^n$, the goal is to create an equivalent ciphertext $\boldsymbol{c}' \in \mathbb{Z}_{q_1}^n$ with a smaller absolute decoding error. The idea is to adjust $\boldsymbol{c}$ by subtracting some $\boldsymbol{\delta_c}$ to a multiple of q_1/q_2, after which $\boldsymbol{c}' = (\boldsymbol{c} - \delta_{\boldsymbol{c}}) \cdot (q_2/q_1) \in \mathbb{Z}_{q_1}^n$. For correctness, and to avoid changing decoding noise too much, the vector $\boldsymbol{\delta_c} \in \mathbb{Z}_{q_1}^n$ must satisfy that $[\langle \boldsymbol{\delta_c}, \boldsymbol{x} \rangle]_{q_1}$ is small and equal to 0 modulo t. It turns out that if $A = [A_1; A_2]$ with $A_2 \in \mathbb{Z}_{q_1}^{n \times n}$ invertible over q, we can use[3]:

$$\boldsymbol{\delta_c} := t \cdot A_2^t \cdot [A_2^{-t} \cdot \boldsymbol{c} \cdot t^{-1}]_{q_1/q_2},$$

3: For small-secret LWE one equivalently has $A_2 = I_n$, which simplifies the expression.

where t^{-1} and A_2^{-t} are the inverses of t and A_2^t modulo q_1/q_2. Finally, given the current and the new modulus q_1, q_2, respectively, and a ciphertext $\boldsymbol{c} \in \mathbb{Z}_{q_1}^n$, the modulus switching is performed as a "round and scale" procedure:

$$\begin{aligned} \text{BGV.ModSwitch} &: \mathcal{C} \times \mathbb{N} \times \mathbb{N} \to \tfrac{q_2}{q_1}\mathcal{C}, \\ \text{BGV.ModSwitch}(\boldsymbol{c}, q_1, q_2) &:= \left[\left((\boldsymbol{c} - \boldsymbol{\delta_c}) \cdot \frac{q_2}{q_1} \right) \right]_{q_2}. \end{aligned} \tag{4.6}$$

By applying this approach, the number of possible sequential multiplications, given a moduli chain $q_1 \approx b^{L+1}$, actually becomes roughly L, since after each multiplication the bound on the noise can be reduced from b^2 back to (approximately) b, while the modulus only decreases by a factor b, preventing an exponential growth [BGV12, Lemma 4].

Key switching. After performing a tensor product between two ciphertexts, in addition to the noise growth, the dimension of the resulting ciphertext increases from n to n^2. Given $\boldsymbol{c}^\times = \boldsymbol{c}_1 \otimes \boldsymbol{c}_2 \in \mathbb{Z}_q^{n^2}$, by the properties of the tensor product, $\boldsymbol{c}^\times$ can still be decrypted using a tensored version of the secret key, namely $\boldsymbol{x} \otimes \boldsymbol{x}$:

$$[\langle \boldsymbol{c}_1 \otimes \boldsymbol{c}_2, \boldsymbol{x} \otimes \boldsymbol{x} \rangle]_q = [\langle \boldsymbol{c}_1, \boldsymbol{x} \rangle \cdot \langle \boldsymbol{c}_2, \boldsymbol{x} \rangle]_q.$$

Nevertheless, this strategy is not sustainable in the long term, as the sizes of the ciphertexts and keys grow exponentially. A solution is to perform a *key switching* procedure. Given a

ciphertext $\boldsymbol{c}$, encrypted under a secret key $\boldsymbol{x}_1$, the objective is to find a new ciphertext $\boldsymbol{c}'$, under a new secret key $\boldsymbol{x}_2$, such that decrypt$(\boldsymbol{c}, \boldsymbol{x}_1) =$ decrypt$(\boldsymbol{c}', \boldsymbol{x}_2)$. In order to enable this procedure, the key generator must produce a *key switching key* $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ that will be publicly shared. The abstract idea is that this new key is some sort of an encryption of the old secret key $\boldsymbol{x}_1$ under $\boldsymbol{x}_2$, and because $\boldsymbol{c}$ is public we can compute an encryption of $[\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle]_q$ under $\boldsymbol{x}_2$ using only homomorphic scalings and additions. Note that the main difference with a full bootstrapping procedure is that there is no further removal of the decryption noise, allowing for a much simpler procedure.

Given two secret keys $\boldsymbol{x}_1, \boldsymbol{x}_2$, their respective dimensions n_1, n_2 and the current modulus q, the key generation procedure for key switching first generates a matrix $A_{\text{ks}} \in \mathbb{Z}_q^{n_1 \times n_2}$ following the public key generation procedure with the secret key $\boldsymbol{x}_2$. While generated by the public key generation procedure, A_{ks} is not made public and should be seen as a matrix containing n_2 Regev encryptions [Reg05] of zero, one for each row. To conclude the key switching key generation phase, the first secret key $\boldsymbol{x}_1$ is added component-wise in the first column of A_{ks}. The final matrix is, considering the addition over the first column, equal to $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2} = A_{\text{ks}} + \boldsymbol{x}_1 \in \mathbb{Z}_q^{n_1 \times n_2}$. Since A_{ks} is uniform-looking by LWE, the secret $\boldsymbol{x}_1$ is hard to be recovered from $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$.

At this point, the key switching can be performed by multiplying the (transposed) matrix $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ with the ciphertext $\boldsymbol{c}^\times$. Intuitively, the presence of $\boldsymbol{x}_1$ in the first column of $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ decrypts the message contained in the first coordinate of $\boldsymbol{c}^\times$, but at the same time this message is re-encrypted by adding to the latter the new encryptions of zero under $\boldsymbol{x}_2$:

$$\begin{aligned}\text{keySwitch}(\boldsymbol{c}, \tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}) &:= (\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2})^t \cdot \boldsymbol{c} \\ &\equiv ([\langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle]_q, 0, \ldots, 0) + (A_{\text{ks}})^t \cdot \boldsymbol{c} \in \mathbb{Z}_q^{n_2}.\end{aligned}$$

We observe that applying this approach as is might not produce a valid ciphertext. The problem is that $\tau^t_{\boldsymbol{x}_1 \to \boldsymbol{x}_2} \cdot \boldsymbol{c} \in \mathbb{Z}_q^{n_2}$ produces a new encryption of the old message by adding n_1 fresh ciphertexts to it, each scaled by c_i. As each c_i can be as large as $q/2$ the result will contain a significant amount of noise that easily exceeds the $q/2$ threshold. There are different approaches to tackle this issue: BV [BV11], GHS and HYBRID [GHS12b].

We will stick to BV: it consists in decomposing the ciphertext $\boldsymbol{c}$ as a longer binary vector using $\texttt{Bits}(\boldsymbol{c})$ and decomposing

[BV11]: Brakerski and Vaikuntanathan, *'Efficient Fully Homomorphic Encryption from (Standard) LWE'* . In: FOCS '11

[GHS12b]: Gentry, Halevi, and Smart, *'Homomorphic Evaluation of the AES Circuit'* . In: CRYPTO '12

the secret key $\boldsymbol{x}_1$ as $\texttt{Pow}(\boldsymbol{x}_1)$; this implies choosing $n_1 = n^2 \log(q)$. The noise contained in the resulting ciphertext will be smaller, since the encryptions of 0 contained in A_{ks} will be multiplied with a vector of smaller magnitude, that is $\texttt{Bits}(\boldsymbol{c})$; the partial decryption will still be valid since $\langle \texttt{Bits}(\boldsymbol{c}), \texttt{Pow}(\boldsymbol{x}_1) \rangle = \langle \boldsymbol{c}, \boldsymbol{x}_1 \rangle$. We formalize the two main procedures as follows:

$$\begin{aligned} \text{BGV.KeySwitchGen} &: \mathbb{Z}_q^{n_1} \to \mathbb{Z}_q^{n_1 \times n_2}, \\ \text{BGV.KeySwitchGen}(\boldsymbol{x}_1) &= A_{\text{ks}} + (\texttt{Pow}(\boldsymbol{x}_1), 0, \ldots, 0) \in \mathbb{Z}_q^{n_1 \times n_2}, \end{aligned}$$

where $A_{\text{ks}} \in \mathbb{Z}_q^{n_1 \times n_2}$ is a new public key under the new secret $\boldsymbol{s}_2$, and the addition is performed in the first column.

$$\begin{aligned} \text{BGV.KeySwitch} &: \mathbb{Z}_q^{n_1} \to \mathbb{Z}_q^{n_2}, \\ \text{BGV.KeySwitch}(\boldsymbol{c}) &:= \texttt{Bits}(\boldsymbol{c}) \cdot (\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2})^t \in \mathbb{Z}_q^{n_2}, \end{aligned}$$

where $\tau_{\boldsymbol{x}_1 \to \boldsymbol{x}_2}$ is the output of BGV.KeySwitchGen($\boldsymbol{x}_1$).

4.2.3 Original LWE-based GSW cryptosystem

One year after the introduction of the BGV cryptosystem, the Gentry-Sahai-Waters (GSW) [GSW13] scheme was proposed. Initially, it was presented as a "conceptually simple" alternative to the already available HE schemes. Today it is considered a fundamental building block to bootstrap advanced constructions such as TFHE. In particular, homomorphic operations are performed "naturally" between ciphertexts as they are encoded in matrices – note that we will follow the notation from [AP14], where ciphertexts are not square matrices, but the two representations are semantically equivalent. As a consequence, multiplications are performed as matrix multiplications without requiring any key switching procedure. The noise growth is reduced by encoding the message to be encrypted in a so-called gadget matrix G; a multiplication between two ciphertexts C_1 and C_2 is indeed evaluated as $C_1 \cdot G^{-1}(C_2)$. The GSW cryptosystem is defined by the LWE dimension and modulus n, q and by the number of samples $m = n \log(q)$. We assume q to be some power of two. Additionally, we define χ as some small distribution over the integers. A plaintext message is set to be in $\{0, 1\} \subset \mathbb{Z}$.

[GSW13]: Gentry, Sahai, and Waters, *'Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based'* . In: CRYPTO '13

[AP14]: Alperin-Sheriff and Peikert, *'Faster Bootstrapping with Polynomial Error'* . In: CRYPTO '14

The key generation phase is exactly the same as the one in the BGV cryptosystem, we thus have $\boldsymbol{x} = (1, -\boldsymbol{x}') \in \mathbb{Z}_q^n$ as the secret key and $A := \left[\begin{array}{c|c} \boldsymbol{b} & B \end{array} \right] \in \mathbb{Z}_q^{m \times n}$, with $\boldsymbol{b} := B\boldsymbol{x}' + \boldsymbol{s} \in$

$\mathbb{Z}_q^m$. For simplicity, we will consider a binary message space $\mathcal{M} := \{0,1\}$ and ciphertext space $\mathcal{C} := \mathbb{Z}_q^{n\times m}$.

Gadget decompositions. Gadget matrices were firstly (implicitly) introduced by Ajtai [Ajt96] and formalized by Micciancio and Peikert [MP12] (although referred to as primitive matrices) as lattice trapdoors. Refer to [GMP19, Def. 4] for a precise definition; for our needs, we only require the basic $\mathbb{Z}_q$-gadget matrix of size $w = 2$, constructed as $\texttt{Pow}(1) \otimes I_n$, where $\texttt{Pow}(1) = (1, 2, 4, \ldots, 2^{\log(q)-1})$. As a result, we obtain the following block matrix $G \in \mathbb{Z}_q^{n\times m}$.

[Ajt96]: Ajtai, *'Generating hard instances of lattice problems (extended abstract)'* . In: STOC '96

[MP12]: Micciancio and Peikert, *'Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller'* . In: EUROCRYPT '12

$$G = \begin{pmatrix} 1\,2\,4\ldots 2^{\log(q)-1} & & \\ & \ddots & \\ & & 1\,2\,4\ldots 2^{\log(q)-1} \end{pmatrix}. \quad (4.7)$$

By abuse of notation, we define $G^{-1} : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_2^{n\log(q)\times m}$ to be the application of $\texttt{Bit}$ over each entry of the input matrix. We will not further discuss gadget (randomized) decompositions, but we refer the interested reader to [GMP19]. The main operations in GSW are defines as follows:

[GMP19]: Genise, Micciancio, and Polyakov, *'Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More'* . In: EUROCRYPT '19

- $\text{Enc}_{pk}(\mu)$: the encryption function under a fixed public key pk is defined as:

$$\begin{aligned} &\text{GSW.Enc}_{pk} : \mathcal{M} \to \mathcal{C}, \\ &\text{GSW.Enc}_{pk}(\mu) := [A^t R + G\mu]_q \in \mathcal{C}, \end{aligned} \quad (4.8)$$

  where $\mu \in \mathcal{M}$ and $R \sim \{0,1\}^{m\times n}$ is uniformly sampled. Each column can be seen as a BGV encryption of zero with $t = 1$.
- $\text{Dec}_{sk}(C)$: the decryption function under a fixed secret key sk is defined as

$$\begin{aligned} &\text{GSW.Dec}_{sk} : \mathcal{C} \to \mathcal{M}, \\ &\text{GSW.Dec}_{sk}(\boldsymbol{c}) := \lfloor [\langle \boldsymbol{c}_j, \boldsymbol{x}\rangle]_q \rceil_2 \in \mathcal{M}, \end{aligned} \quad (4.9)$$

  where $\boldsymbol{c}_j$ is the j-th column of C, with $j = \log(q)$, where $\mu \cdot q/2$ has been added in the first component.
- $\text{Add}(C, C)$: to homomorphically add two underlying messages, evaluate an addition between their corresponding ciphertexts:

$$\begin{aligned} &\text{GSW.Add} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}, \\ &\text{GSW.Add}(C, C') := C + C' \in \mathcal{C}. \end{aligned} \quad (4.10)$$

- Mul(C, C'): to homomorphically multiply two underlying messages, evaluate the following:

$$\begin{aligned} &\text{GSW.Mul} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}, \\ &\text{GSW.Mul}(C, C') := C \cdot G^{-1}(C') \in \mathcal{C}. \end{aligned} \tag{4.11}$$

Encoding the message in the gadget matrix has the advantage of allowing for an asymmetric and quasi-additive error growth in multiplications, we refer to [AP14, Section 3] for further details.

4.3 A lattice framework for HE

Until now, we simply described the HE schemes without considering a lattice perspective. Now we provide a geometric intuition of the processes involved, in lattice terminology, when using an LWE-based cryptosystem. Then, we generalize these to any lattice.

4.3.1 Interpreting LWE from a lattice perspective.

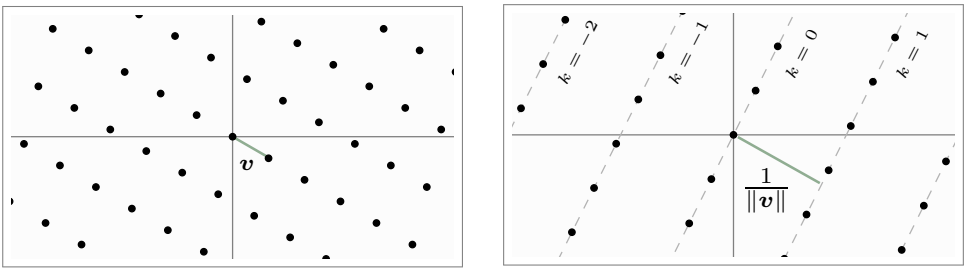
In LWE-based cryptosystems, the hardness assumption is used twice: for the key generation and for the encryption phases. Let us start from a geometric point of view: LWE has a reduction to hard problems over (random) q-ary lattices. More precisely, let $B \in \mathbb{Z}_q^{m \times n}$, we consider the q-ary lattice $\mathcal{L}_q(B)$ (Definition 27), and we call B a *q-ary basis* of $\mathcal{L}_q(B)$. Note that the lattice $\mathcal{L}_q(B)$ is generated by the columns of B and the *q-vectors* $q\boldsymbol{e}_i \in q\mathbb{Z}^m$ where $\boldsymbol{e}_i$ is a unit vector. The q-dual lattice of $\mathcal{L}_q(B)$, also known as a parity check or SIS lattice, is given by $\mathcal{L}_q^{\perp}(B)$ (Definition 28).

While for $\mathcal{L}_q(B)$ we already mentioned a generating set of $m + n$ vectors, writing down a basis for these lattices can be a bit more involved. Typically, we can write $B^t = [B_1^t \mid B_2^t]$ where $B_2 \in \mathbb{Z}_q^{n \times n}$ is invertible, in this case $\mathcal{L}_q(B)$ and $\mathcal{L}_q^{\perp}(B)$ have a *lattice basis* of the *normal form*

$$\begin{pmatrix} qI_{m-n} & B_1 B_2^{-1} \\ 0 & I_n \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & I_{m-n} \\ qI_n & -B_2^{-t} B_1^t \end{pmatrix}, \tag{4.12}$$

respectively.

When $B \in \mathbb{Z}_q^{m \times n}$ is uniformly random we call $\mathcal{L}_q(B)$ a random q-ary lattice (of parameter n, m, q). Note that if $\mathcal{L}_q(B)$ is a



(a) $\mathcal{L}$ and its shortest vector $\boldsymbol{v}$

(b) $\mathcal{L}^*$ with partitions given by $\boldsymbol{v}$

Figure 63: The relation between short vectors in $\mathcal{L}$ and partitions in $\mathcal{L}^*$

random q-ary lattice, then its q-dual $\mathcal{L}_q^\perp(B)$ is also a random q-ary lattice[4].

For the key generation, the LWE assumption is used to safely hide a short vector in such a random q-ary lattice. To see this, let us consider an LWE instance $(B, \boldsymbol{b} = B\boldsymbol{x}' + \boldsymbol{s} \bmod q)$, where $B \in \mathbb{Z}_q^{m\times(n-1)}$ and $\boldsymbol{x}' \in \mathbb{Z}_q^{n-1}$ are uniformly random, and $\boldsymbol{s} \in \mathbb{Z}_q^m$ is a short vector sampled from some distribution $s_i \sim \chi$. Firstly, note that $\boldsymbol{b}$ (when lifted to $\mathbb{Z}^m$) is in fact a BDD instance in the random q-ary lattice $\mathcal{L}_q(B)$, as, up to addition by q-vectors, $\boldsymbol{b}$ has distance only $\|\boldsymbol{s}\|$ to the lattice point $B\boldsymbol{x}'$. While this interpretation will be of interest later, we will turn this BDD instance into a planted short vector, focusing on the key generation. By the LWE assumption the vector $\boldsymbol{b} \in \mathbb{Z}_q^m$ is indistinguishable from a uniformly random vector, and thus in particular we can append it as a column to B to construct $A := \left[\begin{array}{c|c} \boldsymbol{b} & B \end{array}\right] \in \mathbb{Z}_q^{m\times n}$, that is indistinguishable from a random q-ary basis.

The lattice $\mathcal{L}_q(A)$ with q-ary basis A is thus indistinguishable from a random q-ary lattice, however, the knowledge of the secret $\boldsymbol{x} := (1, -\boldsymbol{x}')$, allows one to find the *secret short vector* $\boldsymbol{s} \in \mathcal{L}_q(A)$ by computing $A \cdot \boldsymbol{x} \equiv \boldsymbol{s} - B \cdot \boldsymbol{x}' \equiv \boldsymbol{s} \in \mathbb{Z}_q^m$.

4: Technically, there is a difference in distributions which can be avoided by restricting prime q and full-rank matrices $B \in \mathbb{Z}_q^{m\times n}$. We will ignore this minor difference.

4.3.2 Dual vectors and partitions.

We briefly recall some concepts introduced in Section 2.1.5 Recall that the dual lattice $\mathcal{L}^*$ consists of all vectors having an integer inner product with all primal vectors $\boldsymbol{v} \in \mathcal{L}$. As a result, any primal vector $\boldsymbol{v} \in \mathcal{L}$ naturally partitions the dual lattice $\mathcal{L}^*$ into layers $\{\boldsymbol{y} \in \mathcal{L}^* : \langle \boldsymbol{v}, \boldsymbol{y}\rangle = k\}$, for $k \in \mathbb{Z}$, orthogonal to $\boldsymbol{v}$, as illustrated in Figure 63. Of course, since $(\mathcal{L}^*)^* = \mathcal{L}$, this also applies in the opposite direction: any dual vector partitions the primal lattice. The distance between

these layers is $1/\|\boldsymbol{v}\|$; the shorter the vector, the larger the separation. More generally, there is a strong relation between the covering radius of the dual lattice and the shortest vector in the primal, and vice versa [Ban93].

[Ban93]: Banaszczyk (1993), *'New bounds in some transference theorems in the geometry of numbers'* . In: Mathematische Annalen

By considering the shortest vector in the primal, one can generate layers of points in the dual lattice that are at the maximum distance between each other. This space is precisely where a message can be encoded and hidden.

Let us consider a short vector $\boldsymbol{v} \in \mathcal{L}$ and a BDD instance $\boldsymbol{t} = \boldsymbol{y} + \boldsymbol{e} \in \mathbb{R}^m$ where $\boldsymbol{y} \in \mathcal{L}^*$ and $\boldsymbol{e} \in \mathbb{R}^m$ is small. Note that, given the target $\boldsymbol{t}$, it is generally hard to recover the error $\boldsymbol{e}$. However, since we know that $\langle \boldsymbol{y}, \boldsymbol{v} \rangle$ is integer, we can, knowing $\boldsymbol{v} \in \mathcal{L}$, compute

$$\langle \boldsymbol{e}, \boldsymbol{v} \rangle \equiv \langle \boldsymbol{t}, \boldsymbol{v} \rangle \bmod 1 \in [-\tfrac{1}{2}, \tfrac{1}{2}).$$

At this point, if the distance $1/\|\boldsymbol{v}\|$ between the partitions is large and the error $\boldsymbol{e}$ is small, one can hope that the closest partition to $\boldsymbol{t}$ is the one with the lattice point $\boldsymbol{y}$, or equivalently, $|\langle \boldsymbol{e}, \boldsymbol{v} \rangle| < \frac{1}{2}$. In particular we have $|\langle \boldsymbol{e}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{v}\|$. If this is the case, we do not solely recover $\langle \boldsymbol{e}, \boldsymbol{v} \rangle$ modulo 1, but we recover the exact value $\langle \boldsymbol{e}, \boldsymbol{v} \rangle \in \mathbb{R}$, and therefore gain partial information about the error $\boldsymbol{e}$. We can therefore try to encode the message in this error so that one can recover it from this partial information. Moreover, note that we only have interest in the error here and thus might as well consider only the syndrome $\boldsymbol{t} \equiv \boldsymbol{e} \in \mathbb{R}^m / \mathcal{L}^*$.

Note that in the LWE setting, one considers the q-dual $q\mathcal{L}^* = \mathcal{L}_q^\perp(A)$ of $\mathcal{L} = \mathcal{L}_q(A)$ and thus instead of considering the previous inner products modulo 1, one has to consider them modulo q. Because $\mathcal{L}_q^\perp(A) \subset \mathbb{Z}^m$ this conveniently gives an integer space for the syndromes and allows one to consider the inner products modulo q as elements of $\mathbb{Z}_q$.

4.3.3 Visualizing an LWE ciphertext.

As already spoiled in the previous subsection, an LWE ciphertext can be interpreted as a BDD instance, or more precisely a syndrome, with respect of the q-dual lattice $\mathcal{L}_q^\perp(A)$ of the trapdoor lattice $\mathcal{L}_q(A)$. To observe this we consider a BGV ciphertext, which (for now, assume a message $\mu = 0$) is sampled as $\boldsymbol{c} := [t \cdot A^t \cdot \boldsymbol{r}]_q \in \mathbb{Z}_q^n$, with small $\boldsymbol{r} \in \{0,1\}^m$ and $t \geq 1$. A GSW ciphertext, for $\mu = 0$ can be considered as a collection of $n \log(q)$ different BGV ciphertexts using $t = 1$. We thus focus on the BGV case.

In LWE terms, the ciphertext is a random subset sum (controlled by $t \cdot \boldsymbol{r}$) of the LWE samples contained in the public key, essentially forming a new LWE sample. We claim however that $[A^t \cdot (t \cdot \boldsymbol{r})]_q$ represents the coset $t \cdot \boldsymbol{r} + \mathcal{L}_q^\perp(A)$, or more precisely the syndrome $t \cdot \boldsymbol{r} \in \mathbb{Z}^m / \mathcal{L}_q^\perp(A)$, where $t \cdot \boldsymbol{r}$ is a small error (refer to Figure. 64).

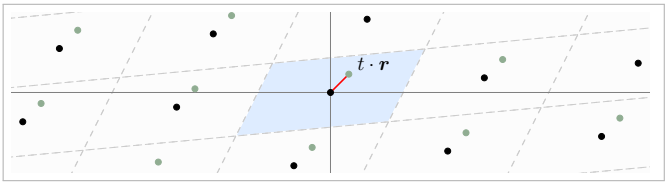


Figure 64: Example of a lattice $\mathcal{L}^*$ (black points) and the coset $t \cdot \boldsymbol{r} + \mathcal{L}^*$ (green)

Indeed, for any $\boldsymbol{y} \in \mathcal{L}_q^\perp(A)$, we have by definition that $A^t \boldsymbol{y} \equiv 0 \bmod q$, and thus

$$A^t(t \cdot \boldsymbol{r} + \boldsymbol{y}) \equiv A^t \boldsymbol{y} + A^t(t \cdot \boldsymbol{r}) \equiv \boldsymbol{0} + A^t(t \cdot \boldsymbol{r}) \equiv A^t(t \cdot \boldsymbol{r}) \bmod q.$$

The LWE assumption precisely says that this syndrome is indistinguishable from a uniformly random syndrome in the (seemingly) random q-ary lattice $\mathcal{L}_q^\perp(A)$. For the decryption procedure, consider the secret short vector $\boldsymbol{s} \equiv A\boldsymbol{x} \in \mathcal{L}_q(A)$. Note that BGV first computes the modulo-q inner product $[\langle \boldsymbol{c}, \boldsymbol{x} \rangle]_q$. Again, we claim that this is precisely the inner product between the short primal vector $\boldsymbol{s}$, partitioning $\mathcal{L}_q^\perp(A)$, and the syndrome error $t \cdot \boldsymbol{r} \in \mathbb{Z}^m / \mathcal{L}_q^\perp(A)$. Indeed

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle \equiv \langle A^t \cdot (t \cdot \boldsymbol{r}), \boldsymbol{x} \rangle \equiv \langle t \cdot \boldsymbol{r}, A \cdot \boldsymbol{x} \rangle \equiv \langle t \cdot \boldsymbol{r}, \boldsymbol{s} \rangle \bmod q, \quad (4.13)$$

and assuming that the error $t \cdot \boldsymbol{r}$ and the short vector $\boldsymbol{s} \in \mathcal{L}_q(A)$ are small enough, i.e., $|\langle t \cdot \boldsymbol{r}, \boldsymbol{s} \rangle| < q/2$, we obtain precisely the inner product $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle \in \mathbb{Z}$.

Since $t \cdot \langle \boldsymbol{r}, \boldsymbol{s} \rangle \equiv 0 \bmod t$, BGV encodes a message $\mu \in \{0, \ldots, t-1\}$ by adding $\mu \boldsymbol{e}_1$ to $\boldsymbol{c}$, using that $\langle \mu \boldsymbol{e}_1, \boldsymbol{x} \rangle = \mu \equiv \mu \bmod t$. Essentially, this corresponds to the addition of some vector $\mu \cdot \boldsymbol{d}$ to $t \cdot \boldsymbol{r}$ for which we know that $\langle \boldsymbol{d}, \boldsymbol{s} \rangle$ is small and satisfies $\langle \boldsymbol{d}, \boldsymbol{s} \rangle \equiv 1 \bmod t$. More details on this *decoding by short primal vector* process are given in Section 4.5.1.1.

4.3.4 Generalizing key generation: a (trapdoor) lattice

We have seen that the key generation of LWE-based encryption schemes constructs a *public* lattice $\mathcal{L}$ along with some short

secret vector $\boldsymbol{s} \in \mathcal{L}$. This will be the starting point of our generalization.

For LWE, the lattice $\mathcal{L}$ is a (seemingly) random q-ary lattice $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$ of dimension m along with a planted short vector. In our framework we will allow $\mathcal{L}$ to be any lattice with a planted short vector $\boldsymbol{s} \in \mathcal{L}$, and we leave the specifics of this to the instantiation of our framework. We assume that the public lattice $\mathcal{L}$ is represented by some (long) basis B, and that the short vector $\boldsymbol{s} \in \mathcal{L}$ is secret. The basis B will be our public key and the short vector $\boldsymbol{s}$ the secret key. As all the other steps will be generic to any lattice, one essentially only has to change the instantiation of the key generation.

The LWE-based encryption schemes we discussed encrypt by constructing a syndrome decoding instance in the q-dual lattice $q\mathcal{L}^*$ of $\mathcal{L}$. This requires some discretized ambient space $\mathcal{D} \supset q\mathcal{L}^*$ for the error and decoding targets, which we will consider in more detail later. For decryption, the short primal vector gives a partition of the q-dual lattice as seen in Figure 63, which allows to partially decode or distinguish the syndrome (or the message).

Definition 62 (*Lattice key generation*)

Keygen is a probabilistic algorithm that on input a security parameter λ generates $pk, sk := (B, D), \boldsymbol{s} \leftarrow$ Keygen(1^λ) such that

- *$\mathcal{L} := \mathcal{L}(B)$ and $\mathcal{D} := \mathcal{L}(D)$ are lattices satisfying $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$,*
- *(short vector) $\boldsymbol{s} \in \mathcal{L}$,*
- *$\langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \equiv 1 \bmod t$, where $\boldsymbol{d}_1$ is the first column of D,*

for some integer parameters $q \geq 2, t \geq 1$ and $m := \dim(\mathcal{L}) \geq 2$.

For security reasons, the decryption procedure must be hard given only the public basis B of $\mathcal{L}$. Several choices can be made here. For LWE-based schemes one usually uses the decision-LWE assumption which, with our interpretation, essentially says that a BDD syndrome in a random q-ary lattice is indistinguishable from a uniformly random syndrome.

Assumption 1 (*SDP is indistinguishable from uniform*)

Let $(B, D), \boldsymbol{s} \leftarrow$ Keygen(1^λ), $\sigma_{enc} \geq \|\tilde{D}\| \cdot \sqrt{\ln(2m+4)/\pi}$ and $C := B^{-t}$. The $\Delta\text{SDP}_{C,D,\sigma_{enc}}$ problem asks to distinguish with non-negligible advantage in λ between two distributions of syndromes $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{poly(\dim(C))} \in \mathcal{P}(qC) \cap \mathcal{D}$: either they are SDP

instances of the form

$$\boldsymbol{c}_i \leftarrow \boldsymbol{e}_i \bmod \mathcal{P}(qC) \in \mathcal{P}(qC) \cap \mathcal{D} \text{ with error } \boldsymbol{e}_i \leftarrow \mathcal{G}_{\mathcal{D},\sigma_{enc}},$$

or they are all uniform. The assumption is that the $\mathrm{SDP}_{C,D,\sigma_{enc}}$ *problem is infeasible.*

Later, we will show that our Lattice-BGV and Lattice-GSW schemes, to be presented in Section 4.5, are IND-CPA secure under Assumption 1.

Remark 6

The choice for an unstructured lattice $\mathcal{L}$ *and a single secret short vector* $\boldsymbol{s} \in \mathcal{L}$ *was made to closely follow the classical LWE-based HE schemes, and to present the framework in its simplest form. One could easily replace this by allowing multiple short vectors, or even a dense sublattice* $S \subset \mathcal{L}$ *with a specific decoding procedure in its dual* S^**. This could for instance improve the error resistance and the number of messages that can be packed in a single ciphertext. Moreover, one could simply replace the lattices here by structured (module-)lattices and the main steps of our approach should still work with minor changes.*

4.3.5 The ambient space

Let $\mathcal{L}$ and $q\mathcal{L}^*$ be our primal and q-dual lattice, respectively, as generated by the key generation procedure, and let B be the public basis of $\mathcal{L}$. Recall that in LWE-based schemes, encryptions are BDD instances $\boldsymbol{t} = \boldsymbol{v} + \boldsymbol{e} \in \mathbb{Z}^m$, or more precisely, their syndrome $\boldsymbol{t} \bmod q\mathcal{L}^* \in \mathbb{Z}^m/q\mathcal{L}^* \cong \mathbb{Z}_q^n$. Therefore, these BDD instances, and their errors, live in the ambient space $\mathbb{Z}^m$, which $q\mathcal{L}^*$ is naturally contained in as it is a q-ary lattice. Having $\mathbb{Z}^m$ as an ambient space is useful, as we know how to sample small errors $\boldsymbol{e} \in \mathbb{Z}^m$, needed for the public-key encryptions, and because the syndrome space $\mathbb{Z}^m/q\mathcal{L}^*$ is finite and naturally has a nice group structure to make the homomorphic operations work.

Generally however, we do not have that $q\mathcal{L}^*$ lies in $\mathbb{Z}^m$, as we do not restrict ourselves merely to cases were $q\mathcal{L}^*$ is an integer lattice. We thus have to construct some appropriate ambient space $\mathcal{D} \supset q\mathcal{L}^*$ with similar properties. We therefore aim to construct a generalized ambient space that satisfies the following two properties:

1. It has a good public basis D that allows to sample small errors $\boldsymbol{e} \in \mathcal{D}$.

2. It satisfies $q\mathcal{D} \subset q\mathcal{L}^* \subset \mathcal{D}$.

The first point generalizes that the integer lattice $\mathbb{Z}^m$ has an orthogonal basis which allows to efficiently sample small errors. The second point is a generalization of the q-ary property, and forces $\mathcal{D}/q\mathcal{L}^*$ to be a subgroup of $\mathbb{Z}_q^m$, which will later be important to make the homomorphic operations work.

Simple ambient space. To sample a small error $\boldsymbol{e} \in \mathcal{D}$ form a Discrete Gaussian distribution we require that the Gram-Schmidt norm $\|\tilde{D}\|$ of the ambient space basis D is sufficiently small compared q. The simplest way to obtain an appropriate ambient space $\mathcal{D}$ with a sufficiently short basis is to simply pick $\mathcal{D} := \mathcal{L}^*$ with $D := B^{-t}$. For q sufficiently large $\mathcal{D} = \mathcal{L}^*$ is dense enough compared to $q\mathcal{L}^*$ and the basis B^{-t} is sufficiently small compared to that of $q\mathcal{L}^*$ to sample relatively small errors. This choice also trivially satisfies the condition $q\mathcal{D} \subset q\mathcal{L}^* \subset \mathcal{D}$.

Ciphertext compression. While the previous choice of $\mathcal{D} := \mathcal{L}^*$ works, it creates a rather large ambient space. This is reflected in the size of the ciphertext syndromes which are elements in $\mathcal{D}/q\mathcal{L}^*$ and thus require $\log_2(|\mathcal{D}/q\mathcal{L}^*|)$ bits to write down, which with the simple choice above gives $m \log_2(q)$. For GSW ciphertexts, which require multiple syndromes, this even grows to $\log_2(|\mathcal{D}/q\mathcal{L}^*|)^2$ bits.

Recall that, in the case of LWE, we have an ambient space $\mathcal{D}$ such that $\mathcal{D}/q\mathcal{L}^* \cong \mathbb{Z}_q^n$ and thus we only require $n \log_2(q) < m \log_2(q)$ bits for the syndrome. In general we can also construct such a smaller ambient space. The idea is that while $D = B^{-t}$ is an appropriate choice, the resulting Gram-Schmidt vectors $\boldsymbol{d}_1^*, \ldots, \boldsymbol{d}_m^*$ typically wildly vary in size while we only require to bound their maximum value $\max_i \|\boldsymbol{d}_i^*\|$. Instead of picking $D = B^{-t}$ we can instead scale the different basis vectors $\boldsymbol{d}_i$ by some appropriate positive integer factors $p \mid q$, thereby increasing $\|\boldsymbol{d}_i^*\|$ by a factor p while decreasing $|\mathcal{D}/q\mathcal{L}^*|$ by a factor p. This works especially well if we assume that $q = p^k$ for some small modulus p. For more details about ciphertext compression we refer to Section 4.4. We also explain there how this can be considered as a generalisation of small-secret LWE.

Coefficient representation. So far, to keep the geometric intuition, we have presented our ambient space $\mathcal{D}$ as a lattice with a good basis D, and small errors $\boldsymbol{e} \in \mathcal{D}$ and syndromes

$\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ are simply assumed to be represented by lattice vectors. One of the crucial aspects of the key switching procedure using the `Bits` and `Pow` functions in BGV and the gadget matrix used in the GSW scheme is however that we are working with integer vectors (or say in $\mathbb{Z}_q^m$). Furthermore, working with such integer vectors is more useful in concrete implementations. This is, however, not a problem in our more general setting, as in fact a lattice vector $\boldsymbol{v} \in \mathcal{D}$ can still be represented by their integer coefficient vector $\boldsymbol{y} \in \mathbb{Z}^m$ in the basis D such that $\boldsymbol{v} = D \cdot \boldsymbol{y}$, or in a basis C of $\mathcal{L}^*$ (using that $\mathcal{D} \subset \mathcal{L}^*$). Similarly, the isomorphism $\mathcal{D}/q\mathcal{L}^* \cong G \subset \mathbb{Z}_q^m$ where G is a subgroup of $\mathbb{Z}_q^m$ can be made explicit by the map $\boldsymbol{c} \mapsto C^{-1}\boldsymbol{c} \bmod q \in G \subset \mathbb{Z}_q^m$. Later we will see that applying the `Bit`, `Pow` and gadget matrix can simply be applied to this coefficient representation. E.g. for $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ and $\boldsymbol{s} \in \mathcal{L}$ we have that $C^{-1}\boldsymbol{c} \in \mathbb{Z}_q^m$ and $C^t\boldsymbol{s} \in \mathbb{Z}^m$, and we have that

$$\langle \boldsymbol{c}, \boldsymbol{s} \rangle \equiv \langle C^{-t}\boldsymbol{c}, C^t\boldsymbol{s} \rangle \equiv \langle \mathtt{Bits}(C^{-t}\boldsymbol{c}), \mathtt{Pow}(C^t\boldsymbol{s}) \rangle \bmod q.$$

Remark 7

Multiplication by C^{-1} gives an explicit isomorphism from $\mathcal{D}/q\mathcal{L}^*$ into a subgroup of $\mathbb{Z}_q^m$. Noting that $C^{-1} = B^t$ we can recognize that BGV and GSW are also using this representation (see $A^t \cdot \boldsymbol{r}$ in the encryption). Another explicit isomorphism into some finite abelian group $G' = \mathbb{Z}^m/\mathcal{R}$ is given through multiplication by D^{-1}. In the special case that C, D differ by a matrix in Hermite Normal Form (as is the case for our constructions), the map D^{-1} gives an explicit isomorphism into $G' = \prod_{i=1}^m \mathbb{Z}/p_i\mathbb{Z}$ for some $p_i \mid q$.

4.4 Ciphertext compression and small-secret LWE

We have explained in Sections 4.3.4 and 4.3.5 that we require an ambient space $\mathcal{D}$ satisfying $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ along with a basis D of $\mathcal{D}$ for which $\|\tilde{D}\|$ is sufficiently bounded. The first property is required for the Homomorphic operations to work, while the second property is required to sample sufficiently small errors.

A basic choice is $\mathcal{D} := \mathcal{L}^*$ with $D := B^{-t}$ derived from the public basis B of $\mathcal{L}$. Then the ciphertext space equals $\mathcal{D}/q\mathcal{L}^* \cong \mathbb{Z}_q^m$ which can be rather large. This gives larger

function ConstructAmbientSpace (C, p, k)
  Input: A basis C of some lattice $\mathcal{L}^* \subset \mathbb{R}^m$, and moduli p, $q = p^k$ for some $k > 0$.
  Output: A basis D such that $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$ and $\max_i \|\tilde{\mathbf{d}}_i\| \leq \max_i \|\tilde{\mathbf{c}}_i\|$.
  Compute the GSO $\tilde{C}$ of C
  $q \leftarrow p^k$, $\ell_{\max} \leftarrow \max_i \|\tilde{\mathbf{c}}_i\|$
  for $i \leftarrow 1$ **to** m **do**
    $j_i = \min\{k, \lfloor \log_p(\ell_{\max}/\|\tilde{\mathbf{c}}_i\|) \rfloor\}$ for all $i = 1, \ldots, m$
    $\mathbf{d}_i \leftarrow p^{j_i} \cdot \mathbf{c}_i$ for all $i = 1, \ldots, m$
  return $D = (\mathbf{d}_1, \ldots, \mathbf{d}_m)$

Algorithm 4: Ambient space construction

ciphertexts but also an additional overhead for both the additions and multiplications. In this section we consider a general compression technique that can be used to obtain a sparser ambient space, while maintaining the same bound on $\|\tilde{D}\|$ as the approach above.

4.4.1 Compressed ambient space

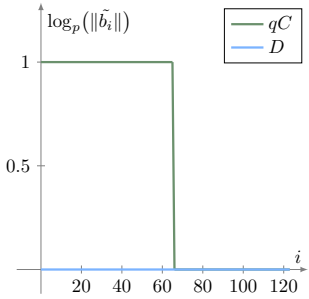
Let $C := B^{-t}$ be the basis of $\mathcal{L}^*$ dual to B, so that qC is a basis of $q\mathcal{L}^*$. Before presenting our generalized ambient space construction we make a further observation about LWE-based schemes, where we assume that the basis qC of $\mathcal{L}_q^\perp(A)$ has the block structure as in 4.12. Namely, one can observe that the Gram-Schmidt profile $q(\|\tilde{\boldsymbol{c}}_1\|, \ldots, \|\tilde{\boldsymbol{c}}_m\|)$ of qC has a Z-shape: it starts with vectors of norm q, and ends with vectors of norm 1.

Now, if one scales down these first q-vectors by q, they would obtain the basis D with vectors $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n, q\boldsymbol{c}_{n+1}, \ldots, q\boldsymbol{c}_m$. Now observe two things: firstly, the Gram-Schmidt profile of D is entirely flat, all Gram-Schmidt vectors have norm 1, and secondly, by construction $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$, in fact D is a basis of the ambient space $\mathbb{Z}^m$. Furthermore, after size-reducing D one precisely obtains a basis of orthogonal unit vectors.

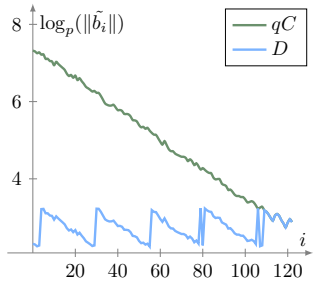
We will now generalize this construction. Generally, the Gram-Schmidt profile of C does not necessarily have such a Z-shape, so more work is needed. Note that so far we have not specified the value of q, thus we still have the freedom to do so. Let us therefore assume that q is of the form $q = p^k$ for some small prime p. Instead of scaling vectors by q (or 1) only, we will allow any scaling of the form p^i for $0 \leq i \leq k$. This choice is made for simplicity, for example we could also allow q to be a

product of many small primes, or another value that works well for the specific basis we have as input.

To construct such an ambient space $\mathcal{D}$, we will use the procedure given in Algorithm 4 that generates an appropriate $\mathcal{D}$ such that $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ along with a good basis D (see Figure 65).



(a) qC is the public basis of a random q-ary lattice, with $p = q = 123$



(b) qC is an LLL-reduced basis of (a rotation of) $\mathbb{Z}^m$, with $p = 3$ and $q = p^5$

Figure 65: Gram-Schmidt profiles of the bad basis qC of $q\mathcal{L}^*$ and the good basis D of $\mathcal{D}$ as given by Algorithm 4

Lemma 24 (*Compressed ambient space*)

Let C be a basis of some lattice $\mathcal{L}^* \subset \mathbb{R}^m$ and let $p > 0, q = p^k$ be some moduli for some $k > 0$. Let $\ell_{\min} = \min_i \|\tilde{\boldsymbol{c}}_i\|$ and $\ell_{\max} = \max_i \|\tilde{\boldsymbol{c}}_i\|$. Then Algorithm 4 returns a basis D of an ambient space $\mathcal{D}$ such that $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ and such that for all $i = 1, \ldots, m$ we have

$$\min\{\ell_{max}/p, q\ell_{\min}\} \leq \|\tilde{\boldsymbol{d}}_i\| \leq \ell_{\max}.$$

Proof. Consider the variables used in Algorithm 4. Note that each j_i satisfies $0 \leq j_i \leq k$, and thus $1 \leq p^{j_i} \leq q$ for all $i = 1, \ldots, m$. So by construction $q\mathcal{L}^* \subset \mathcal{L}(D) \subset \mathcal{L}^*$. For the bounds note that $\|\tilde{\boldsymbol{d}}_i\| = p^{j_i} \cdot \|\tilde{\boldsymbol{c}}_i\|$. The upper bound then follows immediately from the fact that $j_i \leq \log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|)$. For the lower bound we consider two cases, first suppose that $\log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|) < k + 1$, then $j_i > \log_p(\ell_{\max}/\|\tilde{\boldsymbol{c}}_i\|) - 1$ and thus $\|\tilde{\boldsymbol{d}}_i\| > \ell_{\max}/p$. For the other case we have that $j_i = k$ and we obtain $\|\tilde{\boldsymbol{d}}_i\| = q \cdot \|\tilde{\boldsymbol{c}}_i\| \geq q\ell_{\min}$. □

Connections with structural lattice reduction. Highly related to our ambient space construction is the work by Gama, Izabachene, Nguyen and Xie [Gam+16] on *structural lattice reduction*. This work makes the generalization of LWE to *Group*

[Gam+16]: Gama, Izabachene, P. Q. Nguyen, and Xie, '*Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems*' . In: EUROCRYPT '16

LWE (GLWE); where LWE is a decoding problem in a lattice $\mathcal{L} \subset \mathbb{Z}^m$ and an ambient space $\mathbb{Z}^m$ such that $\mathbb{Z}^m/\mathcal{L} \cong \mathbb{Z}_q^n$, the GLWE problem considers the decoding problem in a lattice $\mathcal{L}$ such that $\mathbb{Z}^m/\mathcal{L} \cong G$ for any finite abelian group G. Furthermore, given a large enough group G, it is shown that for any lattice $\mathcal{L}$ (not necessarily integer) one can construct an ambient space $\mathcal{D}$ with $\mathcal{D}/\mathcal{L} \cong G$ along with a nearly orthogonal basis of $\mathcal{D}$. They use this to show reductions of worst-case lattice problems to GLWE, and they show how to construct a GSW-like scheme based on GLWE a lá [AP14; DM15].

[AP14]: Alperin-Sheriff and Peikert, *'Faster Bootstrapping with Polynomial Error'* . In: CRYPTO '14
[DM15]: Ducas and Micciancio, *'FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second'* . In: EUROCRYPT '15

In this chapter we come across similar problems and generalizations. For one, we also need to construct an appropriate ambient space and construct one in a similar approach as in [Gam+16], with the main difference being that we do not fix the group G in advance, but only restrict it to a subgroup of some $\mathbb{Z}_q^m$. This makes our approach simpler, and a bit more efficient in terms of the parameters we obtain, but we stress that one might as well use structural lattice reduction within our framework if one aims for a certain group structure. Secondly, the generalized modulus switching that we will consider is related to the *group switching* method in [Gam+16]. Again however, we do not need the full generalization and present a simpler method that is sufficient for our construction.

As a main difference between these works is that our goal is to present and explain HE from a *lattice perspective*, while one could view [Gam+16] as presenting it from a group structure or algebraic perspective. Furthermore, while we could instantiate the HE schemes we obtain with (the lattices occurring in) GLWE, our framework is more general and allows to be instantiated with *any lattice* with a sufficiently short secret vector.

4.4.1.1 Connections with short secret LWE

Short secret LWE, firstly introduced in [App+09], refers to a particular definition of LWE where the distribution used for $\boldsymbol{s}$ is also used for the secret $\boldsymbol{x}$. We start from the main result.

[App+09]: Applebaum, Cash, Peikert, and Sahai, *'Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems'* . In: CRYPTO '09

Lemma 25 ([App+09])

There is a polynomial time reduction from short-secret-LWE$_{n,m,\chi,q}$ to LWE$_{n,m,\chi,q}$ and one from LWE$_{n,m,q,\chi}$ to short-secret-LWE$_{n,m+n,\chi,q}$.

So it is clear that, up to an increase in the dimension, the two problems are equivalent. We now want to show that using a short secret in LWE does not change our perspective, as short

secret LWE is simply an explicit way of turning BDD into a SDP, so it only changes the "interpretation". At a high-level, short-secret-LWE can be seen as a compression technique, where one is only interested in maintaining information about the small error $\boldsymbol{s}$, and not the particular "close vector" $A\boldsymbol{x} \bmod q$.

Consider some $A \in \mathbb{Z}_q^{m \times n}$ which is drawn uniformly, where m is sufficiently larger than n. Then, with overwhelming probability, A contains n linearly independent vectors. Hence, we can re-write the matrix A in the form

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \in \mathbb{Z}_q^{m \times n}$$

where $A_2 \in \mathbb{Z}_q^{n \times n}$ consists of the n independent rows, thus is invertible modulo q. Notice that permuting rows does not change the underlying lattice. Now we can multiply A by A_2 to obtain

$$A_2^{-1} \cdot A = \begin{pmatrix} A_2^{-1} A_1 \\ I_n \end{pmatrix}$$

Just like permuting the rows, applying this invertible transformation does not change the underlying lattice (Lemma 11). We can therefore re-state LWE from $\boldsymbol{b} := A\boldsymbol{x} + \boldsymbol{s}$ to

$$\begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} A' \\ I_n \end{pmatrix} \cdot \boldsymbol{x}' + \begin{pmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$

where $A' = A_2^{-1} A_1$ and $\boldsymbol{x}' = A_2 \boldsymbol{x}$. At this point, simply consider

$$L = \begin{pmatrix} I_{m-n} & -A' \\ 0 & I_n \end{pmatrix}$$

which is computable since it only uses public material, and multiply everything by L:

$$L \cdot \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} = L \cdot \begin{pmatrix} A' \\ I_n \end{pmatrix} \cdot \boldsymbol{x}' + L \cdot \begin{pmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$

This leads to

$$\begin{pmatrix} \boldsymbol{b}_1 - A'\boldsymbol{b}_2 \\ \boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{x}' \end{pmatrix} + \begin{pmatrix} \boldsymbol{s}_1 - A'\boldsymbol{s}_2 \\ \boldsymbol{s}_2 \end{pmatrix} \in \mathbb{Z}_q^m$$

And notice that now we have a new short secret LWE sample in the top part, as

$$\underbrace{\boldsymbol{b}_1 - A'\boldsymbol{b}_2}_{\text{new } \boldsymbol{b}} = \underbrace{A'}_{\text{new } A} \cdot \underbrace{-\boldsymbol{s}_2}_{\text{new } \boldsymbol{x}} + \underbrace{\boldsymbol{s}_1}_{\text{new } \boldsymbol{s}}$$

so the new secret key becomes $-\boldsymbol{s}_2$, and if we recover it we

can also find a solution for the original instance with $\boldsymbol{x} = A_2^{-1} \cdot \boldsymbol{x}' = A_2^{-1} \cdot (\boldsymbol{s}_2 - \boldsymbol{b}_2)$. Clearly, the new variables are more compact even though they still contain all the information from before, besides the (typically larger) vector $\boldsymbol{x}$. The lattice $\mathcal{L}$ in which the original LWE/BDD instance lived is unchanged, and can be represented by A' because

$$\mathcal{L} = \begin{pmatrix} A' \\ I_n \end{pmatrix} \cdot \mathbb{Z}^n + q\mathbb{Z}^m.$$

More precisely, $\begin{pmatrix} A' \\ I_n \end{pmatrix}$ and $\begin{pmatrix} qI_{m-n} \\ 0 \end{pmatrix}$ together form a basis of $\mathcal{L}$. We also want to emphasize that the lattice operations one can do with short-secret-LWE can also be achieved with general-LWE, although this might require some rewriting. To summarize, since short-secret-LWE can be seen as (a particular instance of) standard LWE, although with slightly different parameters, our arguments can also be applied to the short-secret-LWE setting with minor changes.

4.5 Abstract HE cryptosystems from lattices

Following the key generation and ambient space construction in the previous section, we now present our lattice-based generalizations of the BGV [BGV12] and GSW [GSW13] homomorphic encryption cryptosystems.

4.5.1 Lattice-BGV

The Lattice-BGV scheme is a lattice generalization of BGV that can in principle be instantiated, in public key mode, using any hardness assumption that hides the secret shortest vector(s) of a lattice (e.g., LWE, NTRU, LIP and so on), and for which decoding in the dual is hard.

The message space and ciphertext space are respectively $\mathcal{M} := \mathbb{Z}_t$ and $\mathcal{C} := \mathcal{D}/q\mathcal{L}^*$ for some plaintext modulus $t \ll q$, lattice $\mathcal{L}$, and ambient space $\mathcal{D}$ obtained from the key generation process from Definition 62, which satisfies $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}$. We assume q to be built as a moduli ladder, i.e., a product of many primes, like in the LWE-based BGV.

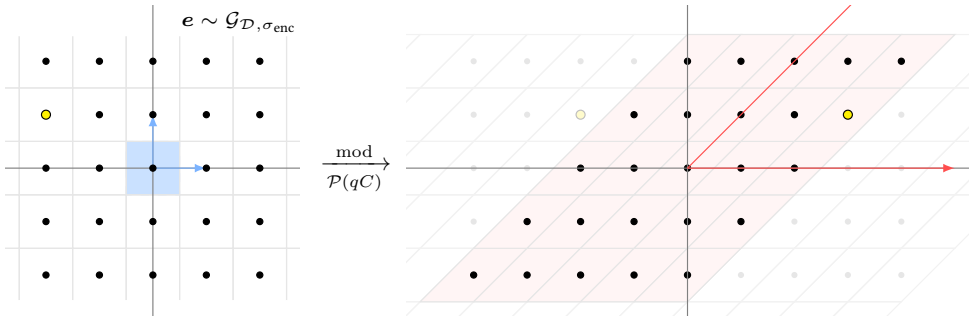


Figure 66: Hiding a sample $e \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ encrypting zero (in yellow) within the fundamental region of $\mathcal{D}/q\mathcal{L}^*$, assuming $\mathcal{L}^* = \mathbb{Z}^n$, with $q = 5$

Encryption and decryption. As introduced in Section 4.3.3, the encryption process in BGV samples a small error $\boldsymbol{e} \in \mathcal{D}$, containing an encoded message, which is then canonically represented by some syndrome $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$. The secret short vector $\boldsymbol{s} \in \mathcal{L}$ will be used to partially decode it and recover the encoded message. To sample a small error in the ambient space $\mathcal{D}$ we precisely use its public good basis D that we constructed in Section 4.3.5. From the key generation we also have that the first basis vector $\boldsymbol{d}_1$ of D satisfies $\langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \equiv 1 \bmod t$. The goal is to generalize the BGV.Enc$_{pk}(\mu)$ procedure in Eq. (4.2). We start by considering a small vector $\boldsymbol{r} \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ in the ambient space $\mathcal{D}$, with σ_{enc} chosen as explained in Section 4.3.5. Then, the message is encoded in the small error $\boldsymbol{e} \in \mathcal{D}$ as follows:

$$\boldsymbol{e} := t \cdot \boldsymbol{r} + D \cdot (\mu, 0, \ldots, 0)^t = t \cdot \boldsymbol{r} + \boldsymbol{d}_1 \mu \in \mathcal{D}, \qquad (4.14)$$

where $\mu \in \mathcal{M}$ is the message, and $\boldsymbol{d}_1$ is the first basis vector of D, the good basis of $\mathcal{D}$. The discrete Gaussian sampler $\mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ returns a point $\boldsymbol{r}$ that is close to the origin in the ambient space $\mathcal{D}$, and thus $\boldsymbol{e} \approx t \cdot \boldsymbol{r}$ is also small. The following step is to hide the sampled error $\boldsymbol{e}$, containing the message μ, in a canonical coset representative $\boldsymbol{c} \in \boldsymbol{e} + q\mathcal{L}^*$. For this we use the fundamental region of $q\mathcal{L}^*$ defined by the public basis $qC = qB^{-t}$:

$$\boldsymbol{c} = \boldsymbol{e} \bmod \mathcal{P}(qC) \in \mathcal{D}/q\mathcal{L}^*. \qquad (4.15)$$

This process is represented in Figure 66. Because $\mathcal{L}^* \subset \mathcal{D} \subset q\mathcal{L}^*$ one can represent $\boldsymbol{e}$ and $\boldsymbol{c}$ as integer vectors with respect to the basis C and thus the above operation can be simply implemented with modular arithmetic. We now formalize the

just presented procedures.

$$\begin{aligned}&\mathcal{L}\text{-BGV.Enc}_{pk} : \mathcal{M} \to \mathcal{C},\\ &\mathcal{L}\text{-BGV.Enc}_{pk}(\mu) := t\boldsymbol{r} + \boldsymbol{d}_1\mu \bmod \mathcal{P}(qC).\end{aligned} \tag{4.16}$$

Recall that the secret key $\boldsymbol{s}$ is a short vector in the primal lattice $\mathcal{L}$. The decryption therefore follows the arguments from Section 4.3.

$$\begin{aligned}&\mathcal{L}\text{-BGV.Dec}_{sk} : \mathcal{C} \to \mathcal{M},\\ &\mathcal{L}\text{-BGV.Dec}_{sk}(\boldsymbol{c}) := [[\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_q]_t,\end{aligned} \tag{4.17}$$

Decryption is correct, since:

$$[[\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_q]_t \equiv [t \cdot \langle \boldsymbol{r}, \boldsymbol{s}\rangle + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle \cdot \mu]_t \equiv \mu \in \mathbb{Z}_t,$$

assuming that $|t\cdot\langle \boldsymbol{r}, \boldsymbol{s}\rangle + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle\cdot\mu| < q/2$. Notice that $\langle \boldsymbol{d}_1, \boldsymbol{s}\rangle \equiv 1 \bmod t$ by Definition 62. We also define the corresponding homomorphic operations:

$$\begin{aligned}&\mathcal{L}\text{-BGV.Add} : \mathcal{C} \times \mathcal{C} \to \mathcal{C},\\ &\mathcal{L}\text{-BGV.Add}(\boldsymbol{c}, \boldsymbol{c}') := \boldsymbol{c} + \boldsymbol{c}' \in \mathcal{C}.\end{aligned} \tag{4.18}$$

Homomorphic multiplications can be evaluated as in the standard BGV scheme, and their correctness follows from the fact that the inner product of two tensor products becomes the product of two inner products,

$$\begin{aligned}&\mathcal{L}\text{-BGV.Mul} : \mathcal{C} \times \mathcal{C} \to \mathcal{C} \otimes \mathcal{C},\\ &\mathcal{L}\text{-BGV.Mul}(\boldsymbol{c}, \boldsymbol{c}') := \boldsymbol{c} \otimes \boldsymbol{c}' \in \mathcal{C} \otimes \mathcal{C}.\end{aligned} \tag{4.19}$$

Given two ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{D}/q\mathcal{L}^*$, their tensor product $\boldsymbol{c}^\times = \boldsymbol{c}_1 \otimes \boldsymbol{c}_2 \in (\mathcal{D}/q\mathcal{L}^*) \otimes (\mathcal{D}/q\mathcal{L}^*)$ is a ciphertext under the new secret key $\boldsymbol{s}^\times = \boldsymbol{s} \otimes \boldsymbol{s} \in \mathcal{L} \otimes \mathcal{L}$ representing the product of the encrypted values μ_1 and μ_2:

$$\begin{aligned}[[\langle \boldsymbol{c}^\times, \boldsymbol{s}^\times\rangle]_q]_t &\equiv [[\langle \boldsymbol{c}_1 \otimes \boldsymbol{c}_2, \boldsymbol{s} \otimes \boldsymbol{s}\rangle]_q]_t \equiv [[\langle \boldsymbol{c}_1, \boldsymbol{s}\rangle \cdot \langle \boldsymbol{c}_2, \boldsymbol{s}\rangle]_q]_t\\ &\equiv [[(t \cdot \langle \boldsymbol{r}_1, \boldsymbol{s}\rangle + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle \cdot \mu_1) \cdot (t \cdot \langle \boldsymbol{r}_2, \boldsymbol{s}\rangle + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle \cdot \mu_2)]_q]_t\\ &\equiv [[t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle^2\mu_1\mu_2]_q]_t \equiv [t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle^2\mu_1\mu_2]_t \equiv \mu_1 \cdot \mu_2 \in \mathbb{Z}_t,\end{aligned}$$

assuming that $|t \cdot e + \langle \boldsymbol{d}_1, \boldsymbol{s}\rangle^2\mu_1\mu_2| < q/2$ where we use $t \cdot e \in \mathbb{Z}$ to refer to the decryption noise coming from the other terms to ease readability. Just as for the original BGV scheme, the decryption noise $t \cdot e$ of the product is roughly the square of that of the input ciphertexts. Notice that, again, with a right instantiation as discussed in Section 4.3.5, the original decryption noise is sufficiently small relative to q and

thus for q sufficiently large the above multiplication can be correctly decrypted.

Even if the noise is bounded by say b, independently of q, it still is roughly squared after a single multiplication, and therefore grows exponentially on repeated multiplications, allowing us to do at most $\log_b(q)$ of them. Similarly, the dimension of the lattice and the ciphertext space is potentially squared as $\dim(\mathcal{L} \otimes \mathcal{L}) = \dim(\mathcal{L})^2$ and $\dim(\mathcal{C} \otimes \mathcal{C}) = \dim(\mathcal{C})^2$ if $q = p^k$ for some prime p. Just as in the LWE case we can solve these two problems by modulus switching and key switching, respectively.

4.5.1.1 Studying the error growth

Before analyzing more advanced primitives, we want to give some formal definitions of the error $\boldsymbol{e} \in \mathcal{D}$, the decryption error $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle|$ and the correctness of decryption in the Lattice-BGV scheme.

Definition 63 (*Error in a Lattice-BGV ciphertext*)

A ciphertext $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ contains an error $\boldsymbol{e} \in \mathcal{D}$ that is defined recursively depending on the sequence of homomorphic operations applied to it.

- *Base case (fresh encryption). A* fresh ciphertext $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ *is generated by hiding some error $\boldsymbol{e} \in \mathcal{D}$ as in* (4.14)*:*

$$\boldsymbol{e} := t\boldsymbol{r} + \boldsymbol{d}_1\mu \in \mathcal{D}, \quad \textit{where } \boldsymbol{r} \sim \mathcal{G}_{\mathcal{D},\sigma_{enc}}.$$

  we say that $\boldsymbol{c}$ contains (or hides) error $\boldsymbol{e}$, where $\boldsymbol{e} \in \mathcal{D}$ is the error *associated to $\boldsymbol{c}$ as $\boldsymbol{c} \equiv \boldsymbol{e} \bmod \mathcal{D}/q\mathcal{L}^*$.*
- *Recursive step (homomorphic operations). Let $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{D}/q\mathcal{L}^*$ be two independent ciphertexts such that $\boldsymbol{c}_1$ has error $\boldsymbol{e}_1$ and $\boldsymbol{c}_2$ has error $\boldsymbol{e}_2$. Then the error of the resulting ciphertext after each homomorphic operation is defined as:*

$$\begin{aligned} \textit{(Addition)} \quad &= \boldsymbol{e}_1 + \boldsymbol{e}_2 = t(\boldsymbol{r}_1 + \boldsymbol{r}_2) + \boldsymbol{d}_1(\mu_1 + \mu_2), \\ \textit{(Tensor product)} \quad &= \boldsymbol{e}_1 \otimes \boldsymbol{e}_2 = (t\boldsymbol{r}_1 + \boldsymbol{d}_1\mu_1) \otimes (t\boldsymbol{r}_2 + \boldsymbol{d}_1\mu_2), \\ \textit{(Multiplication by } \alpha \in \mathbb{Z}) \quad &= \boldsymbol{e} \cdot \alpha = \alpha t\boldsymbol{r} + \boldsymbol{d}_1(\alpha\mu), \end{aligned}$$

This recursive formulation allows one to compute the error contained in any ciphertext by following the structure of the homomorphic computation tree leading to it. Building upon the error definition, we also define the *decryption error* in a

ciphertext, that is directly defined by the decryption algorithm that uses the short primal vector.

Definition 64 (*Decryption error in a Lattice-BGV ciphertext*)

Given the previously defined error in ciphertext, the decryption error *in a ciphertext* $\boldsymbol{c} \equiv \boldsymbol{e} \bmod \mathcal{D}/q\mathcal{L}^*$ *is measured considering the error with respect to the short primal vector* $\boldsymbol{s} \in \mathcal{L}$ *as* $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle| < \|\boldsymbol{e}\|_2 \cdot \|\boldsymbol{s}\|_2$.

Notice that the decryption error is bounded by both the error contained in a ciphertext and the norm of the short vector used to decode. In other words, the shorter the vector $\boldsymbol{s} \in \mathcal{L}$, the smaller the decryption error.

Decoding by short primal vector(s) and correctness. In all presented schemes, decryption is performed by using the shortest vector in the primal lattice. We explain what is needed for correctness in a more general decoding perspective, to indicate what could potentially be gained from alternative decoding methods. Given the one-dimensional lattice $S = s\mathbb{Z} \subset \mathcal{L}$ that is generated by $\boldsymbol{s} \in \mathcal{L}$, this process can be defined as (i) the BDD target $\boldsymbol{e} \in \mathcal{D} \subset \mathcal{L}^*$ is projected onto span(S) as $\pi_{\boldsymbol{s}}(\boldsymbol{e})$ and (ii) the projected point $\pi_{\boldsymbol{s}}(\boldsymbol{e})$ is rounded to the closest lattice point in the dual $\pi_s(\mathcal{L}^*) = S^* = \boldsymbol{s}/\|\boldsymbol{s}\|^2 \cdot \mathbb{Z}$. Remark that

$$\pi_{\boldsymbol{s}}(\boldsymbol{e}) = \text{proj}_{\boldsymbol{s}}(\boldsymbol{e}) = \langle \boldsymbol{e}, \boldsymbol{s} \rangle \cdot \boldsymbol{s}/\|\boldsymbol{s}\|^2.$$

In other words, the decryption $\langle \boldsymbol{e}, \boldsymbol{s} \rangle$ can be interpreted as "how far along the direction $\boldsymbol{s}$ the error $\boldsymbol{e}$ lies", in lattice terminology, it is the rounding error of $\pi_{\boldsymbol{s}}(\boldsymbol{e})$ in S^*. In general, to round correctly, we need that the error of some target value must be smaller than half the minimum distance $\lambda_1(S^*)$. A ciphertext hiding $\boldsymbol{e}$ is correctly rounded if the norm of its projection $\|\pi_{\boldsymbol{s}}(\boldsymbol{e})\|$ is smaller than half the shortest vector of S^* (also see Figure 63, where $\lambda_1(S^*)$ is the green vector in (b)). Since $\|\pi(\boldsymbol{e})\| \leq \|\boldsymbol{e}\|$, we have that $\|\boldsymbol{e}\| < \lambda_1(S^*)/2$ to be correctly decryptable. The shortest vector of S^* has norm $\lambda_1(S^*) = 1/\|\boldsymbol{s}\|$, and since we usually work with q-duals (see beginning of Section 4.3), $\lambda_1(qS^*) = q/\|\boldsymbol{s}\|$. By putting everything together, we have that $\|\boldsymbol{e}\| < q/(2\|\boldsymbol{s}\|)$ to prove correctness of decoding. Finally, by considering the decryption error, we have that an error $\boldsymbol{e}$ can always be uniquely decoded if:

$$|\langle \pi_s(\boldsymbol{e}), \boldsymbol{s} \rangle| \leq \|\pi_s(\boldsymbol{e})\| \cdot \|\boldsymbol{s}\| \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{s}\| < \frac{q}{2\|\boldsymbol{s}\|} \cdot \|\boldsymbol{s}\| = \frac{q}{2}.$$

Notice that, in our setup, the error $\boldsymbol{e}$ is composed of both some random noise $\boldsymbol{r} \in \mathcal{G}_{\mathcal{L}^*,\sigma_{\text{enc}}}$ and an encoded message $\mu \in \mathbb{Z}_t$ in the first public basis vector $\boldsymbol{d}_1$ of $\mathcal{D}$. In a standard public key scheme encrypting values from $\mathbb{Z}_t$, we prove correctness by showing that the norm of $\boldsymbol{r}$ contained in an error $\boldsymbol{e}$ is such that:

$$\|\boldsymbol{r}\| < \frac{q}{2t} - t\|\boldsymbol{d}_1\|.$$

Conversely, there is a non negligible probability to sample $\boldsymbol{r}$ in such a way that $\|\boldsymbol{r}\| > q/(2t)$ – or equivalently $t\|\boldsymbol{r}\| > q/2$ – changing the congruence modulo t of the underlying message. For instance, if we want to encrypt a message from $\mathbb{Z}_2$, the norm of $\boldsymbol{r}$ can be at most roughly $q/4$. If the plaintext space is $\mathbb{Z}_4$, the norm of $\boldsymbol{r}$ can be at most roughly $q/8$, and so on. Notice that similar arguments can be done to prove correctness in the case of encoding some message $\mu \in \mathbb{Z}_t$ as $\mu \cdot q/t$, without scaling $\boldsymbol{r}$.

In a homomorphic encryption setup, proving actual correctness with overwhelming probability for a ciphertext containing the homomorphic evaluation of some polynomial p is more challenging, and still an open problem, even in the standard LWE setting [Mon+23]. Intuitively, one should verify correctness for fresh ciphertext, and set parameters that allow for the correct decryption of homomorphic operations, following the recursive definition of error in Definition 63.

Lemma 26 (*Correctness of decryption*)

A ciphertext $\boldsymbol{c} \in \mathcal{D}/q\mathcal{L}^*$ containing an error $\boldsymbol{e}$ is correctly decryptable using the short dual vector $\boldsymbol{s} \in \mathcal{L}$ if its error is such that

$$\|\boldsymbol{e}\| < \frac{1}{2}\lambda_1(qS^*) = \frac{q}{2\|\boldsymbol{s}\|},$$

hence, its decryption error is such that

$$|\langle \boldsymbol{e}, \boldsymbol{s}\rangle| \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{s}\| < \left\langle \frac{q}{2\|\boldsymbol{s}\|}, \|\boldsymbol{s}\| \right\rangle = \frac{q}{2}$$

This guarantees unique and correct decoding.

For instance, for a fresh error $\boldsymbol{e} \equiv t\boldsymbol{r} + \boldsymbol{d}_1\mu$, it must be such that $\|\boldsymbol{r}\| < q/(2t) - \|\boldsymbol{d}_1\|t$, so σ_{enc} should be set accordingly.

Average cases and upper bounds for errors. We will show how large decryption errors are on average in fresh ciphertexts and how they can be upper bounded with overwhelming probability. Similar results for additions and multiplications

can be easily obtained following their corresponding error definitions. For multiplications, simply consider the property $\|\boldsymbol{c}_1 \otimes \boldsymbol{c}_2\|_2^2 = \|\boldsymbol{c}_1\|_2^2 \|\boldsymbol{c}_2\|_2^2$, so the norm of some tensor product is simply the product of the norms of the two multiplied vectors.

Lemma 27 (*Fresh decryption error, on average*)

The average amount of decryption error in a fresh ciphertext in $\mathcal{D}/q\mathcal{L}^*$ is smaller than

$$(t \cdot \sigma_{enc})(\sqrt{m} + 1/2) \cdot \|\boldsymbol{s}\|_2$$

with $\boldsymbol{s}$ being the shortest vector $\boldsymbol{s} \in \mathcal{L}$ used to decrypt, and $m := \dim(\mathcal{L}) \geq 2$.

Proof. The decryption error is computed as $\langle \boldsymbol{e}, \boldsymbol{s} \rangle$, therefore

$$\langle \boldsymbol{e}, \boldsymbol{s} \rangle \equiv \langle t \cdot \boldsymbol{r} + D \cdot (\mu, 0, \ldots, 0), \boldsymbol{s} \rangle \bmod \mathcal{D}/q\mathcal{L}^*$$

Notice that $\boldsymbol{s} \in \mathcal{L}^*$ can be considered as a constant, and we have $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle| \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{s}\|$, so we will focus on $\boldsymbol{e}$. The latter is composed of two terms:

- $(t \cdot \boldsymbol{r})$: observe that if $\boldsymbol{r} \sim \mathcal{G}_{\mathcal{L},\sigma_{\text{enc}}}$ then $\|\boldsymbol{r}\|_2$ is on average bounded by $\sqrt{m} \cdot \sigma_{\text{enc}}$, so this term is on average $t \cdot \sqrt{m} \cdot \sigma_{\text{enc}}$
- $(D \cdot (\mu, 0, \ldots, 0))$: this term can be rewritten as $\boldsymbol{d}_1 \cdot \mu$. Observe that, in order to efficiently sample (Lemma 8) from D, we need σ_{enc} to be such that $\sigma_{\text{enc}} > \|\boldsymbol{d}_1\| = \|\boldsymbol{d}_1^*\|$, therefore this can be bounded as $\sigma_{\text{enc}} \cdot (t/2)$ by assuming that, on average, $\mu = t/2$.

Notice though that the euclidean norm is not linear (i.e., $\mathbb{E}[\|\boldsymbol{a} + \boldsymbol{b}\|_2] \neq \mathbb{E}[\|\boldsymbol{a}\|_2] + \mathbb{E}[\|\boldsymbol{b}\|_2]$), however observe that $\mathbb{E}[\|\boldsymbol{a} + \boldsymbol{b}\|_2] < \mathbb{E}[\|\boldsymbol{a}\|_2] + \mathbb{E}[\|\boldsymbol{b}\|_2]$, so we can sum them to have a (conservative) average result. The result follows by refactoring $(t \cdot \sqrt{m} \cdot \sigma_{\text{enc}}) + (t \cdot \sigma_{\text{enc}})/2$. □

Corollary 7 (*Fresh decryption error, upper bound*)

The amount of decryption error in a fresh ciphertext is at most

$$(t \cdot \sigma_{enc})(\sqrt{m} + \sqrt{2\ln(2)} + 1/2) \cdot \|\boldsymbol{s}\|_2$$

with $\boldsymbol{s}$ being the shortest vector $\boldsymbol{s} \in \mathcal{L}$ used to decrypt, and $m := \dim(\mathcal{L}) \geq 2$.

By considering the discrete Gaussian $\mathcal{G}_{\mathcal{L},\sigma_{\text{enc}}}$ as a subgaussian,

we can use the following tail bound:

$$\Pr[\|X\| \geq \sigma_{\text{enc}}\sqrt{m}(1+\varepsilon)] \leq e^{-\varepsilon^2 m/2}, \quad 0 < \varepsilon < 1.$$

We set $e^{-\varepsilon^2 m/2} = 2^{-m}$, with $m \in \mathbb{Z}^+$, to obtain $\varepsilon = \sqrt{2\ln 2}$. Therefore, by extending the proof of Lemma 27, we can move from an average value of

$$(t \cdot \sigma_{\text{enc}})(\sqrt{m} + 1/2) \cdot \|\boldsymbol{s}\|_2,$$

to, with overwhelming probability 2^m, the following upper bound:

$$(t \cdot \sigma_{\text{enc}})(\sqrt{m} + \sqrt{2\ln 2} + 1/2) \cdot \|\boldsymbol{s}\|_2.$$

□

Both estimates can be improved by up to a factor $\sqrt{m}$ when σ_{enc} is above the smoothing parameter of $\mathcal{D}$ (for a sufficiently small ϵ). In this case we can argue that $\langle \boldsymbol{r}, \boldsymbol{s} \rangle$ is negligible close to a 1-dimensional Discrete Gaussian on $\mathbb{Z}$ with parameter $\sigma_{\text{enc}} \cdot \|\boldsymbol{s}\|$ from which a simple tailbound follows.

For simplicity these bounds use the worst-case inequality $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle| \leq \|\boldsymbol{e}\| \cdot \|\boldsymbol{s}\|$, which is typically off by a factor $\Theta(\sqrt{m})$. Both could be formally improved by a more involved argument when σ_{Enc} is sufficiently above the smoothing parameter of $\mathcal{D}$. Depending on the parametrization we do not always have this guarantee and thus opted for these simpler bounds.

Following usual noise estimations done for BGV, we can set encryption parameters such that the modulus switching reduces the decryption error $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle|$ after a multiplication back to the noise of a fresh encryption. Then, by setting $q \geq \|\langle \boldsymbol{e}, \boldsymbol{s} \rangle\|_2^{\ell+1}$, with $\boldsymbol{e}$ being the error in a fresh ciphertext, this would allow for the evaluation of (roughly) up to ℓ multiplications.

A concrete option could be to pick $q = \Omega((t \cdot \sigma_{\text{enc}})(\sqrt{n} + \sqrt{2\ln(2)} + 1/2) \cdot \|\boldsymbol{s}\|_2)^{\ell+1}$, from Corollary 7 which allows for ℓ multiplications, followed by modulus switching, with overwhelming probability depending on m. Notice that of course this is a worst-case bound that is not really practical. Just like in LWE, performance are improved by using short keys, hence it seems natural in our case to use lattices that admit extremely short vectors $\boldsymbol{s} \in \mathcal{L}$, which in turn would allow to use smaller moduli and parameters. Notice that Algorithm 4 allows to construct an ambient space with respect to arbitrary modulus $q = p^k$ and lattice $\mathcal{L}$, so this is feasible in our setup

and security can be improved by setting a sufficiently large dimension n, so that the BDD targets can not be decoded trivially.

4.5.1.2 Modulus switching as syndrome decoding

We shortly recall the modulus switching step in the original BGV scheme. Given starting and final moduli q_1, q_2 such that $q_2 \mid q_1$ and $q_1 \equiv q_2 \equiv 1 \bmod t$, the LWE modulus switching proceeds in two steps. First the ciphertext is shifted by some $\boldsymbol{\delta} \in \mathbb{Z}^m/q_1\mathcal{L}^* \cong \mathbb{Z}_q^n$ term, with the objective to make it divisible by q_1/q_2, after which it is divided by this factor.

The $\boldsymbol{\delta}$ term may add a small amount of noise, but the subsequent scaling will drastically reduce it. More precisely, given a ciphertext $\boldsymbol{c}_1 \in \mathbb{Z}^m/q_1\mathcal{L}^*$, the modulus switching procedure first moves $\boldsymbol{c}_1$, and therefore the error, by some $\boldsymbol{\delta}$ term such that

$$\boldsymbol{c}' := (\boldsymbol{c} - \boldsymbol{\delta}) \in q_1/q_2 \cdot \mathbb{Z}^m/q_1\mathcal{L}^* \cong q_1/q_2 \cdot \mathbb{Z}_{q_1}^n.$$

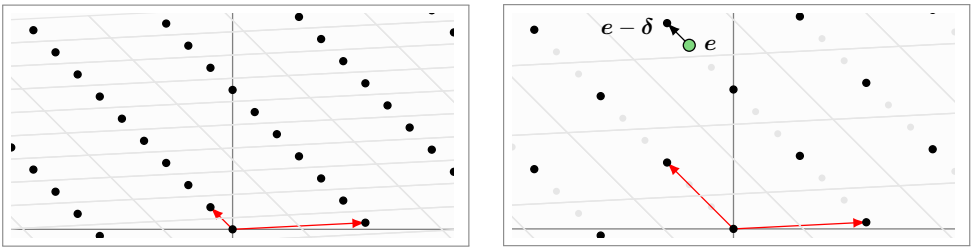
Then, given $\boldsymbol{c}' \in q_1/q_2 \cdot \mathbb{Z}_{q_1}^n$ we divide to obtain the new ciphertext $\boldsymbol{c}_2 := \frac{q_2}{q_1}\boldsymbol{c}' \in \mathbb{Z}_{q_2}^n \cong \mathbb{Z}^m/q_2\mathcal{L}^*$ which is interpreted as a syndrome with respect to $q_2\mathcal{L}^*$. Note that in terms for the corresponding BDD errors $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{Z}^m$, we similarly have $\boldsymbol{e}_2 = (\boldsymbol{e}_1 - \boldsymbol{\delta})/(q_1/q_2)$, and thus the absolute norm $\|\boldsymbol{e}_2\| \approx q_2/q_1 \cdot \|\boldsymbol{e}_1\|$ is significantly reduced, assuming $\boldsymbol{\delta}$ is sufficiently small. In the original BGV scheme this small term is defined as:

$$\boldsymbol{\delta} = t \cdot [\boldsymbol{c} \cdot t^{-1}]_{q_1/q_2} \in \mathbb{Z}_{q_1}^n \cong \mathbb{Z}^m/q_1\mathcal{L}^*,$$

where t^{-1} is the inverse of $t \bmod q_1/q_2$, and it is required to keep the encoding of the message during the scale procedure. Notice that this definition respects the requirements for smallness ($\boldsymbol{\delta}$ is small), scalability (the ciphertext becomes divisible by q_1/q_2), and correctness (the encrypted message is kept) since:

- Each coefficient of $\boldsymbol{\delta}$ is bounded by $\frac{1}{2}t \cdot q_1/q_2$.
- $\boldsymbol{c} - \boldsymbol{\delta} \equiv \mathbf{0} \bmod q_1/q_2$ allows to make the ciphertext divisible by q_1/q_2.
- $\boldsymbol{\delta} \equiv \mathbf{0} \bmod t$ allows to keep the encoding of the message.

Towards a general modulus switching. Having a clear view and more freedom in terms of lattices allows us to define a somehow more generic approach to modulus switching. In



(a) An ambient space $\mathcal{D}_1$

(b) Rounding the error in $q_1/q_2\mathcal{D}_2$

Figure 67: Defining a sparser sublattice $q_1/q_2\mathcal{D}_2 \subset \mathcal{D}_1$ from $\mathcal{D}_1$, assuming $q_1/q_2 = 3$ and $\mathcal{D}_1 = \mathcal{D}_2$ for simplicity. Notice that all the points in $q_1/q_2\mathcal{D}_2$ are divisible by q_1/q_2 by definition

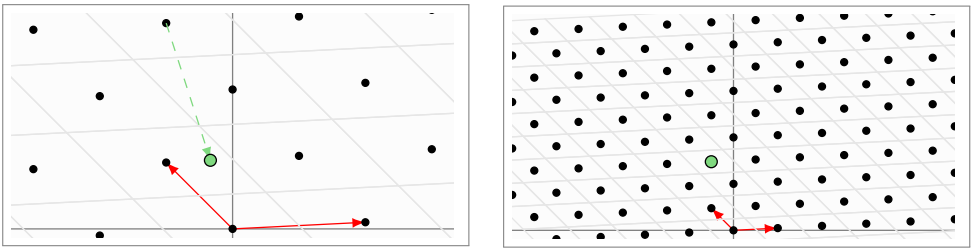
this more general setting, modulus switching has the goal to switch from some initial modulus q_1 to a smaller modulus q_2 such that $q_2 \mid q_1$. In particular, given $q_1\mathcal{L}^* \subset \mathcal{D}_1$ and $q_2\mathcal{L}^* \subset \mathcal{D}_2$ with their respective ambient spaces, the aim is to switch a ciphertext $\boldsymbol{c}_1 \in \mathcal{D}_1/q_1\mathcal{L}^*$ to a ciphertext $\boldsymbol{c}_2 \in \mathcal{D}_2/q_2\mathcal{L}^*$, while reducing the error by roughly a factor q_1/q_2 and while keeping the encoded message intact. We assume that $\mathcal{D}_1$ and $\mathcal{D}_2$ are both generated by our ambient space construction using the moduli q_1 and q_2 respectively. Note that by construction we have that $q_1/q_2 \cdot \mathcal{D}_2 \subset \mathcal{D}_1 \subset \mathcal{D}_2$, so $(q_1/q_2)\mathcal{D}_2$ is a sparsification of $\mathcal{D}_1$ (see Figure 67).

Similarly to the original LWE case, we proceed in two steps which can be summarized by the following:

$$\mathcal{D}_1/q_1\mathcal{L}^* \xrightarrow[\text{round}]{-\boldsymbol{\delta}} (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^* \xrightarrow[\text{scale}]{\times q_2/q_1} \mathcal{D}_2/q_2\mathcal{L}^*,$$

where in the first step we *round* to the sparsification $(q_1/q_2)\mathcal{D}_2$ of the ambient space $\mathcal{D}_1$, and then *scale* down to the new ambient space $\mathcal{D}_2$ with respect to the lattice $q_2\mathcal{L}^*$. We first focus on the rounding part, where, given some ciphertext $\boldsymbol{c}_1 \in \mathcal{D}_1/q_1\mathcal{L}^*$, we need to find some small $\boldsymbol{\delta}$ such that $\boldsymbol{c}_1 - \boldsymbol{\delta} \in (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^*$. Note that this is precisely an (approximate) syndrome decoding problem in $\mathcal{D}_1/(q_1/q_2)\mathcal{D}_2$, where $\boldsymbol{\delta}$ is a small error in the coset $\boldsymbol{c}_1 + (q_1/q_2)\mathcal{D}_2/q_1\mathcal{L}^*$. In particular, the better we can decode in $(q_1/q_2)\mathcal{D}_2$, the smaller $\boldsymbol{\delta}$ can be.

To decode we can again use the good basis that our ambient space $\mathcal{D}_2$, and thus $(q_1/q_2)\mathcal{D}_2$ has, generalizing the LWE setting where decoding in $(q_1/q_2)\mathbb{Z}^m$ is also possible due to a good (orthogonal) basis. In particular, we can use Babai's decoding algorithm which always finds a $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|^2 \leq \frac{q_1}{4q_2}\sum_{i=1}^{m}\|\tilde{\boldsymbol{d}}_i\|^2$, where the latter are the Gram-Schmidt norms of the basis D, which are bounded independently of q by



(a) Scaling the error

(b) The scaled error in $\mathcal{D}_2$

Figure 68: Given a "rounded" ciphertext in $q_1/q_2\mathcal{D}_2$, we can scale it down by the factor q_1/q_2

Lemma 24. The dependency on q_1/q_2 is natural as it directly determines the sparsity of $q_1/q_2\mathcal{D}_2$, and that factor will be removed by the scaling step.

Now that the $\boldsymbol{c} - \boldsymbol{\delta}$ point lies in $q_1/q_2\mathcal{D}_2 \subset \mathcal{D}_1$, it is possible to scale it down, along with the lattice $q_1\mathcal{L}^*$, to obtain a new ciphertext with a smaller error (Figure 68) with respect to the lattice $q_1\mathcal{L}^*/(q_1/q_2) = q_2\mathcal{L}^*$.

As a result, we obtain a point that, although scaled from the original lattice $q_1\mathcal{L}^*$ to $q_2\mathcal{L}^*$, has an error that, in absolute terms, is smaller. Ignoring the additional error introduced by the by $\boldsymbol{\delta}$, the reduction is by a factor q_1/q_2.

Despite this nice procedure, we still require caution as this operation could erase the content of the message stored in the first component. In particular, the decoding algorithm could return a point that decrypts to a different message than the one originally contained in $\boldsymbol{c}$. In LWE, this is tackled by using a $\boldsymbol{\delta}$ term that is a multiple of the plaintext modulus t, so that during decryption can easily be removed.

The difference is how the rounding is performed: in LWE, the delta to the closest vector in the sparse lattice is given by $t \cdot [\boldsymbol{c} \cdot t^{-1}]_{q_1/q_2}$, which implicitly acts as a syndrome decoding algorithm in $t\mathcal{D}_1/(q_1/q_2)\mathcal{D}_2 \cong \mathbb{Z}^n_{q_1/q_2}$, returning a small $\delta \in t\mathcal{D}_1$. Mimicking this, combined with our more general syndrome decoding approach, we can therefore compute the $\boldsymbol{\delta}$ term as follows:

$$\boldsymbol{\delta} = t \cdot \text{approxSDP}(\boldsymbol{c} \cdot t^{-1}, (q_1/q_2)\mathcal{D}_2) \in t\mathcal{D}_1. \tag{4.20}$$

Finally, we can formalize the general modulus switching

procedure

$$\mathcal{L}\text{-BGV.GeneralModSwitch} : \mathcal{D}_1 \times \mathbb{N} \times \mathbb{N} \to \mathcal{D}_2,$$
$$\mathcal{L}\text{-BGV.GeneralModSwitch}(\boldsymbol{c}, q_1, q_2) := \left[\left((\boldsymbol{c} - \boldsymbol{\delta}) \cdot \frac{q_2}{q_1}\right)\right]_{q_2}.$$

Lastly, we introduce a lemma that shows how the decryption error in a ciphertext actually decreases under appropriate parameters.

Lemma 28 (*Correctness of general modulus switching*)

Let $q_1 > q_2 > t$ be positive integers such that $q_1 \equiv q_2 \equiv 1 \bmod t$. Let $\mathcal{D}_1$ and $\mathcal{D}_2 \subset (q_2/q_1)\mathcal{D}_1$ be ambient spaces satisfying Definition 62. Given a $\mathcal{L}$-BGV ciphertext $\boldsymbol{c}_1 \in \mathcal{D}_1/q_1\mathcal{L}_1^*$, the $\mathcal{L}$-BGV.GeneralModSwitch$(\boldsymbol{c}_1, q_1, q_2)$ function returns $\boldsymbol{c}_2 \in \mathcal{D}_2/q_2\mathcal{L}^*$ such that

- *(Correctness):* $[\langle \boldsymbol{c}_1, \boldsymbol{s}\rangle]_{q_1} \equiv [\langle \boldsymbol{c}_2, \boldsymbol{s}\rangle]_{q_2} \bmod t$
- *(Decryption error reduction):* $|[\langle \boldsymbol{c}_2, \boldsymbol{s}\rangle]_{q_2}| \leq (q_2/q_1)|\,[\langle \boldsymbol{c}_1, \boldsymbol{s}\rangle]_{q_2}\,| + \rho\|\boldsymbol{s}\|$

where ρ is the approximate decoding radius of the efficient decoder approxSDP on $\mathcal{D}_2$. Using a basis D_2 of $\mathcal{D}_2$ we have $\rho \leq \frac{1}{2}\sqrt{\sum_{i=1}^m \|\tilde{\boldsymbol{d}}_i\|^2}$, where $(\tilde{\boldsymbol{d}}_i)_i = \tilde{D}_2$.

Proof. For correctness, observe that by construction that $\boldsymbol{\delta} \equiv 1 \bmod t$ and $q_2/q_1 \equiv 1 \bmod t$, so we obtain:

$$[\langle \boldsymbol{c}', \boldsymbol{s}\rangle]_{q_2} \equiv \frac{q_2}{q_1}[\langle \boldsymbol{c}+\boldsymbol{\delta}, \boldsymbol{s}\rangle]_{q_1} \equiv [\langle \boldsymbol{c}, \boldsymbol{s}\rangle + \langle \boldsymbol{\delta}, \boldsymbol{s}\rangle]_{q_1} \equiv [\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_{q_1} \bmod t$$

To see that the decryption error decreases, let us consider the decryption of an output of the $\mathcal{L}$-BGV.GeneralModSwitch function. In particular, we evaluate

$$\frac{q_2}{q_1}\langle (\boldsymbol{c} - \boldsymbol{\delta}), \boldsymbol{s}\rangle \equiv \frac{q_2}{q_1}(\langle \boldsymbol{e}, \boldsymbol{s}\rangle - \langle \boldsymbol{\delta}, \boldsymbol{s}\rangle) \bmod q_1$$

From the approximate decoder on $\mathcal{D}_2$ we have that $\|\boldsymbol{\delta}\| \leq \frac{q_1}{q_2} \cdot \rho$. Notice that by using Babai's decoding algorithm we can bound $\|\boldsymbol{\delta}\|$ as $\|\boldsymbol{\delta}\|^2 \leq q_1/(4q_2)\sum_{i=1}^m \|\tilde{\boldsymbol{d}}_i\|^2$, where $\boldsymbol{d}_i$ is the i-th basis vector for $\mathcal{D}_2$. We can therefore always assume that $\rho \leq \frac{1}{2}\sqrt{\sum_{i=1}^m \|\tilde{\boldsymbol{d}}_i\|^2}$, and we bound $q_2/q_1 \cdot |\langle \boldsymbol{e}, \boldsymbol{s}\rangle - \langle \boldsymbol{\delta}, \boldsymbol{s}\rangle|$ as follows:

$$\begin{aligned}[|\langle \boldsymbol{c}', \boldsymbol{s}\rangle]_{q_1}| = \frac{q_2}{q_1}|\langle \boldsymbol{e}, \boldsymbol{s}\rangle - \langle \boldsymbol{\delta}, \boldsymbol{s}\rangle| &\leq \frac{q_2}{q_1}|\langle \boldsymbol{e}, \boldsymbol{s}\rangle| + \frac{q_2}{q_1}\|\boldsymbol{\delta}\| \cdot \|\boldsymbol{s}\| \\ &\leq \frac{q_2}{q_1}|\langle \boldsymbol{e}, \boldsymbol{s}\rangle| + \rho \cdot \|\boldsymbol{s}\| = \frac{q_2}{q_1}[|\langle \boldsymbol{c}, \boldsymbol{s}\rangle]_{q_1}| + \rho\|\boldsymbol{s}\|\end{aligned}$$

so if $\rho\|\boldsymbol{s}\|$ is small compared to $|\langle \boldsymbol{e}, \boldsymbol{s}\rangle|$, the decryption error decreases as it gets scaled by $q_2/q_1 < 1$. □

The bounds could be improved by either a heuristic average-case argument, or by provable bounds by replacing the decoding step by Discrete Gaussian Sampling over $\mathcal{D}_2$. In theory, the approximate decoding radius ρ can be as low as the covering radius $\nu(\mathcal{D}_2) \leq \rho$ of $\mathcal{D}_2$. For LWE we have $\mathcal{D}_2 = \mathbb{Z}^m$ and $\rho = \nu(\mathcal{D}_2) = \frac{1}{2}\sqrt{n}$, which from the lattice perspective has an optimal covering radius up to a constant factor relative to the normalised volume of the lattice. In theory, using lower bounds on the covering radius, a better lattice could improve ρ by a factor up to $\approx \sqrt{\pi e/2} \approx 2.066$.

Key switching. Given two public lattices $\mathcal{L}_1, \mathcal{L}_2$ with secret short vectors $\boldsymbol{s}_1, \boldsymbol{s}_2$, dimensions $m_1, m_2 \in \mathbb{N}$ and ambient spaces $\mathcal{D}_1, \mathcal{D}_2$ for $q\mathcal{L}_1^*, q\mathcal{L}_2^*$ respectively, the goal of the key switching procedure is to transform a ciphertext $\boldsymbol{c} \in \mathcal{D}_1/q\mathcal{L}_1^*$, with respect to the secret key $\boldsymbol{s}_1$, to a ciphertext in $\mathcal{D}_2/q\mathcal{L}_2^*$ containing the same message with respect to the secret key $\boldsymbol{s}_2$. In particular, this allows us to switch from a larger dimensional lattice, say $\mathcal{L}_1 = \mathcal{L} \otimes \mathcal{L}$ obtained after a multiplication, back to the starting dimension. As the key switching procedure can essentially be generalized to any additive homomorphic scheme, without using any specific lattice properties, we will be rather brief about it. We assume without loss of generality that the bases $C = B^{-t}$ and D_1, D_2 of $\mathcal{L}^*$ and $\mathcal{D}_1, \mathcal{D}_2$ respectively differ from each other by an HNF form such that D_i^{-1} gives an explicit map from $\mathcal{D}_i/q\mathcal{L}^*$ to some diagonal group $\prod_{i=1}^m \mathbb{Z}/p^{j_i}\mathbb{Z}$. For simplicity of notation we assume that the `Bits` function on $\mathbb{Z}/p^{j_i}\mathbb{Z}$ is extended to a vector of length $\log_p(q)$ by appending zeros.

Recall from the LWE case that the switching key $\tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2}$ essentially consists of encryptions $D_2(\boldsymbol{c}_i^{(0)} + ((D_1^t \boldsymbol{s}_1)_i, 0, \ldots, 0)^t) \in \mathcal{D}_2/q\mathcal{L}_2^*$ of each coefficient of $D_1^t \boldsymbol{s}_1$ under $\boldsymbol{s}_2$ (ignoring the modulo t encoding), where $D_2\boldsymbol{c}_i^{(0)}$ is an encryption of 0. Given the public ciphertext $\boldsymbol{c} = D_1(c_1, \ldots, c_{m_1})^t \in \mathcal{D}_1/q\mathcal{L}_1^*$ we simply scale and sum over these encryptions to get

$$\begin{aligned}\boldsymbol{c}' &:= D_2 \cdot \sum_{i=1}^{m_1} c_i \cdot \left(\boldsymbol{c}_i^{(0)} + ((D_1^t \boldsymbol{s}_1)_i, 0, \ldots, 0)^t\right) \\ &= \left(\sum_{i=1}^{m_1} c_i \cdot D_2 \boldsymbol{c}_i^{(0)}\right) + D_2 \cdot ([\langle \boldsymbol{c}, \boldsymbol{s}_1\rangle]_q, 0, \ldots, 0).\end{aligned}$$

We thus obtain a scaled sum of encryptions of 0 which masks the message $[\langle \boldsymbol{c}, \boldsymbol{s}_1\rangle]_q$. Assuming $\boldsymbol{c}$ is a valid ciphertext we have $[\langle \boldsymbol{c}, \boldsymbol{s}_1\rangle]_q = te + \mu$ for the message μ and some *small* decryption noise e. Assuming the c_i are small, the errors coming from the sum of encryptions of 0, and the error $D_2 \cdot (te, 0, \ldots, 0)^t$, should be small enough for the new ciphertext $\boldsymbol{c}'$ to correctly decrypt to μ under $\boldsymbol{s}_2$. For the c_i to be small enough we can use the BV binary decomposition trick.

Remark 8

The encryptions of 0 used to construct the switching key in the LWE case are generated using the public key generation procedure. This must be seen as an artifact of the specific key generation procedure under LWE. More generally, this could be interpreted as a secret key encryption procedure, i.e., the valid encryptions are constructed with knowledge of the secret key. This allows to make the error smaller in the direction of $\boldsymbol{s}_2$, while making it uniform in the direction orthogonal to it to compensate for any concrete security loss. One could easily generalize this to our lattice setting, or alternatively just use the public key encryption procedure (albeit leading to a larger decryption error).

Summarized we obtain the following key switching procedure.

1. Sample $m_1 \cdot \lceil \log(q) \rceil$ encryptions $D_2 \cdot \boldsymbol{c}^{(0)}_{(i,j)} \in \mathcal{D}_2/q\mathcal{L}_2^*$ of zero indexed by the pair (i, j) for $i = 1, \ldots, m_1$ and $j = 1, \ldots, \lceil \log(q) \rceil$ via a public or secret key encryption procedure.
2. Compute the switching key $\tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2} = (\boldsymbol{c}^{(0)}_{(i,j)} + (\mathtt{Pow}((D_1^t \boldsymbol{s}_1)_i)_j, 0, \ldots, 0)^t)_{(i,j)}$.
3. To switch from $\boldsymbol{c}$ to $\boldsymbol{c}'$ given $\tau := \tau_{\boldsymbol{s}_1 \to \boldsymbol{s}_2}$ compute

$$\boldsymbol{c}' = D_2 \cdot \sum_{i=1}^{m_1} \sum_{j=1}^{\lceil \log(q) \rceil} (\mathtt{Bit}((D_1^{-t} \boldsymbol{c})_i)_j \cdot \tau_{(i,j)}.$$

4.5.2 Concrete parameters and requirements

Let $C = B^{-t}$ and D be the public bases of $\mathcal{L}^*$ and the ambient space $\mathcal{D}$ respectively. The basis D combined with Lemma 8, allows to sample small errors in $\mathcal{D}$ from a discrete Gaussian with any parameter $\sigma_{\text{enc}} \geq \max \|\tilde{\boldsymbol{d}}_i\| \cdot \sqrt{\ln(2m + 4)/\pi}$. For a small error $\boldsymbol{r}$, sampled from this distribution, the decryption error $\langle \boldsymbol{r}, \boldsymbol{s}\rangle$ with the short secret vector $\boldsymbol{s} \in \mathcal{L}$ will roughly be of size $1/2 \cdot t \cdot \sigma_{\text{enc}} \cdot \|\boldsymbol{s}\|$ (Lemma 27), which we have to compare to the modulus q.

A choice for q would be a value $1 \leq q \leq \max \|\tilde{\boldsymbol{d}}_i\| / \min \|\tilde{\boldsymbol{c}}_i\|$ which guarantees that $\sigma_{\text{enc}} > q \min \|\tilde{\boldsymbol{c}}_i\|/2$, that is a minimum requirement to make the resulting syndrome decoding problem with respect to $q\mathcal{L}^*$ non-trivial to solve[5]. By increasing q we can thus make the lattice $q\mathcal{L}^*$ sparser and allow for a larger decryption error, so naturally one would choose the maximum value for q. Notice that this is also precisely the choice that is made for LWE-schemes, where $q = q/1 = \max \|\tilde{\boldsymbol{d}}_i\| / \min \|\tilde{\boldsymbol{c}}_i\|$.

5: For a secure instantiation, and depending on the precise basis qC of $q\mathcal{L}^*$, one could need a slightly larger margin, for simplicity we will ignore this minor dependency.

We thus observe that a smaller secret vector $\boldsymbol{s} \in \mathcal{L}$ reduces the decryption error while a worse public basis C of $\mathcal{L}^*$ (typically giving larger $\max \|\tilde{\boldsymbol{d}}_i\| \geq \max \|\tilde{\boldsymbol{c}}_i\|$ and smaller $\min \|\tilde{\boldsymbol{c}}_i\|$) increases it but can increase the modulus q even more. Recall that for the BGV scheme the decryption error is roughly squared after a single multiplication, so the minimum requirement on $\|\boldsymbol{s}\|$ would be that $(\|\boldsymbol{s}\| \cdot \sigma_{\text{enc}}) < \mathcal{O}(\sqrt{q})$.

To allow for many homomorphic multiplications, however, the decryption error of fresh ciphertexts should be close to constant with respect to q. In particular, one would preferably have $\|\boldsymbol{s}\| \sim \sigma_{\text{enc}}^{-1} \approx \max \|\tilde{\boldsymbol{d}}_i\|^{-1}$. Notice that for the LWE case we have precisely this, as $\|\boldsymbol{s}\| = \Theta(\sqrt{n})$ and $\max \|\tilde{\boldsymbol{d}}_i\| = 1$, and $\Theta(\sqrt{n}) \ll q$. In the rest of this work we assume to have a small enough secret $\boldsymbol{s} \in \mathcal{L}$ such that we can assume that the decryption error of fresh ciphertexts is sufficiently small relative to the modulus q to perform multiplications. In our instantiation based on LIP (Section 4.6), we will shortly explain how to achieve this.

4.5.3 Lattice-GSW

We can now use previous results to define the Lattice-GSW scheme. The key generation is the same as in the previous scheme: the public key is a bad basis B of a lattice $\mathcal{L}$ along with a good basis D of the ambient space $q\mathcal{L}^* \subset \mathcal{D} \subset \mathcal{L}^*$ where $q = p^k$ for some $p = t \geq 2$, and the secret key $\boldsymbol{s} \in \mathcal{L}$ is a sufficiently short vector of $\mathcal{L}$. The message message space will be $\mathcal{M} := \mathbb{Z}_p$. For the GSW scheme and its operations, it is useful to express elements more explicitly in their coefficient representation. More precisely, we assume that the bases B^{-t} and D of $\mathcal{L}^*$ and $\mathcal{D} \subset \mathcal{L}^*$ respectively differ from each other by a transformation in Hermite Normal Form such that B^t explicitly maps single ciphertext syndromes into a diagonalized coefficient group $\mathcal{D}/q\mathcal{L}^* \to \prod_{i=1}^{m} p^{j_i}\mathbb{Z}/q\mathbb{Z} \subset \mathbb{Z}_q^m$ for some $0 \leq j_i \leq \log_p(q)$. Furthermore we assume that $j_1 = 1$ and thus B^{-t} and D have the same first vector. Both

our ambient space constructions satisfy this property. Let $\ell = \log_p(|\mathcal{D}/q\mathcal{L}^*|) = q \cdot m - \sum_{i=1}^{m} j_i$, then the ciphertext space equals $\mathcal{C} := (\mathcal{D}/q\mathcal{L}^*)^\ell$.

Encryption and decryption. As observed in Section 4.2.3, the encryption process of GSW is slightly different from the BGV one. In particular, a ciphertext will now be a matrix consisting of ℓ encryptions of zero from Lattice-BGV (with $t = 1$), one for each column. So we first define the error matrix E with columns in $\mathcal{D}$ such that:

$$E := \begin{pmatrix} | & | & & | \\ \boldsymbol{e}_1 & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_\ell \\ | & | & & | \end{pmatrix} \text{ where each } \boldsymbol{e}_i \sim \mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}.$$

We identify this matrix space with $\mathcal{D}^\ell$. Then the syndrome matrix $E \bmod q\mathcal{L}^* \in (\mathcal{D}/q\mathcal{L}^*)^\ell$ represents ℓ encryptions of zero (equivalent to $[A^t R]_q$ in LWE-based GSW). In order to encrypt a message $\mu \in \mathbb{Z}_p$, we simply add a multiple $\mu B^{-t} G$ of the dual basis times the gadget matrix to the coefficient representation matrix E. We just have to be careful as the gadget matrix must be constructed with respect to the base p and to match the explicit group structure of $\mathcal{D}/q\mathcal{L}^*$ mentioned above to make sure that $\mu B^{-t} G \in (\mathcal{D}/q\mathcal{L}^*)^\ell$. To do this one should construct G as before, but with $(p^{j_i}, \dots, p^{\log_p(q)-1})$ on row i. The inverse map G^{-1} is also defined as before, noting that one can always write an element in $p^{j_i}\mathbb{Z}/q\mathbb{Z}$ in the basis $p^{j_i}, \dots, p^{\log_p(q)-1}$ and coefficient in $0, \dots, p-1$. Encryption is thus defined as follows:

$$\mathcal{L}\text{-GSW.Enc}_{pk} : \mathcal{M} \to \mathcal{C}, \tag{4.21}$$

$$\mathcal{L}\text{-GSW.Enc}_{pk}(\mu) := E + B^{-t} G \mu \in \mathcal{C}. \tag{4.22}$$

Decryption follows from GSW, it is possible to extract the j-th column of $C \in \mathcal{C}$ corresponding to $\boldsymbol{c}_j = \boldsymbol{e}_j + q/p\boldsymbol{d}_1$, and use the secret key $\boldsymbol{s}$ as follows:

$$\begin{aligned} &\mathcal{L}\text{-GSW.Dec}_{sk} : \mathcal{C} \to \mathcal{M}, \\ &\mathcal{L}\text{-GSW.Dec}_{sk}(C) := \lfloor [\langle \boldsymbol{c}_j, \boldsymbol{s} \rangle]_q \rceil_2 \in \mathcal{M} \end{aligned} \tag{4.23}$$

This procedure is correct, as shown below:

$$[\langle \boldsymbol{c}_j, \boldsymbol{s} \rangle]_q \rceil_p \equiv \lfloor [e + \langle \boldsymbol{d}_1, \boldsymbol{s} \rangle (q/p)\mu]_q \rceil_p \equiv \mu \in \mathcal{M}$$

where, as in Lattice-BGV, $e = \langle \boldsymbol{e}_j, \boldsymbol{s} \rangle$ is small, and we assumed that $\langle \boldsymbol{d}_1, \boldsymbol{s} \rangle \equiv 1 \bmod p$. Notice that e must be roughly smaller

than $q/2p$ to allow for correct decryption. Homomorphic additions immediately follow:

$$\begin{aligned} &\mathcal{L}\text{-GSW.Add} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}, \\ &\mathcal{L}\text{-GSW.Add}(C_1, C_2) := C_1 + C_2 \in \mathcal{C} \end{aligned} \tag{4.24}$$

The multiplication between two ciphertexts almost immediately follows the original GSW one. Given two ciphertexts $C_1, C_2 \in (\mathcal{D}/q\mathcal{L}^*)^\ell$, – encrypting respectively μ_1 and μ_2 – we define the multiplication as $C^\times := C_1 \cdot G^{-1}(B^t C_2)$. Note the transformation $B^t C_2$ to the coefficient representation which is required to apply G^{-1}.

$$\begin{aligned} &\mathcal{L}\text{-GSW.Mul} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}^2, \\ &\mathcal{L}\text{-GSW.Mul}(C_1, C_2) := C_1 \cdot G^{-1}(B^t C_2) \in \mathcal{C}. \end{aligned} \tag{4.25}$$

We now show that the decryption of multiplications is correct, i.e., we obtain a new ciphertext

$$\begin{aligned} C_1^\times &\equiv (C_1 \cdot G^{-1}(B^t C_2)) \\ &\equiv (E_1 + \mu_1 \cdot B^{-t} G) \cdot G^{-1}(B^t C_2) \\ &\equiv E_1 \cdot G^{-1}(B^t C_2) + \mu_1 \cdot C_2 \\ &\equiv (E_1 \cdot G^{-1}(B^t C_2) + \mu_1 \cdot E_2) + (\mu_1 \cdot \mu_2) \cdot B^{-t} G \\ &\equiv E + (\mu_1 \cdot \mu_2) \cdot B^{-t} G \in \mathcal{C}, \end{aligned}$$

that encodes $\mu_1 \cdot \mu_2$ with the new error $E = (E_1 \cdot G^{-1}(B^t C_2) + \mu_1 \cdot E_2)$. As both μ_1 is small, and the coefficients of $G^{-1}(B^t C_2)$ are bounded by $\ell \cdot (p-1)$, the columns of E consist of small combinations of the initial error columns. For appropriate parameters one therefore correctly decrypts the new ciphertext to $\mu_1 \cdot \mu_2 \in \mathcal{M}$.

4.6 A possible instantiation using LIP

When instantiating our Lattice-BGV and Lattice-GSW scheme with an LWE-based key generation we precisely obtain the original BGV and GSW schemes respectively. We now show, as an example to illustrate the flexibility of our framework, a high-level instantiation based on the Lattice Isomorphism Problem. For more details on the CPA-security proof in this setting we refer to Appendix 4.A.1 and 4.A.3.

Key generation. Recall that we only need to instantiate the key generation procedure from Definition 62. We first focus on the lattice and consider the ambient space later. Recall that the key generation procedure should output a public bad description of a primal lattice $\mathcal{L}$ and a sufficiently short secret trapdoor vector $\boldsymbol{s} \in \mathcal{L}$. A LIP instance is defined as:

$$B' = O \cdot B \cdot U \in \mathrm{GL}_m(\mathbb{R})$$

where B, B' are public material, and refer to the basis of the chosen lattice and a public rotation of it, respectively, and $O \in \mathcal{O}_m(\mathbb{R})$ or $U \in \mathrm{GL}_m(\mathbb{Z})$ are the secret transformations. The problem asks to recover either O or equivalently U (up to an automorphism). Notice that it is possible to use any good basis B – generating a lattice with nice decoding properties – even $B = I_m$ to generate a rotation of $\mathbb{Z}^m$, which is what is essentially used by the HAWK scheme [Duc+22]. For now we will simply assume that the first vector $\boldsymbol{b}_1$ in B is a sufficiently short vector, but note that this also easily allows to have multiple short vectors. This fits our setting well, since we naturally have all the required ingredients:

[Duc+22]: Ducas, Postlethwaite, Pulles, and W. v. Woerden, *'Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple'* . In: ASIACRYPT '22

- $B' = O \cdot B \cdot U \in \mathrm{GL}_m(\mathbb{R})$: bad description of the lattice $\mathcal{L} = \mathcal{L}(B')$.
- $\boldsymbol{s} := O \cdot \boldsymbol{b}_1 \in \mathcal{O}_m(\mathbb{R})$: a short vector $\boldsymbol{s} \in O \cdot \mathcal{L}(B) = \mathcal{L}(B')$.

Notice that one can securely sample the basis B' of a certain quality following [DvW22, Algorithm 1], or by applying the procedure to the dual to create a basis C of the dual lattice $\mathcal{L}^* := \mathcal{L}(B')^*$ of a certain quality.

Fixing a concrete set of parameters. The next step is to generate a basis $D \in \mathbb{R}^{m \times m}$ for the ambient space according to Algorithm 4. Ignoring some further technical details we assume that the lattice $\mathcal{L}$ is normalized to have volume 1 and that we can sample a dual basis C of $\mathcal{L}^*$ such that $\|\tilde{C}\| \sim \sigma_{\mathrm{pk}}$ for some sufficiently large parameter $\sigma_{\mathrm{pk}} > 0$. Following Lemma 24 there is a strong dependency between the $\sigma_{\mathrm{pk}} > 0$ value used to sample C, and the allowed modulus q of the scheme. Intuitively, larger values of σ_{pk} correspond to a worse basis, thus in larger spread $\ell_{\max}/\ell_{\min}$ in the Gram-Schmidt norms $\ell_{\min} = \min_i \|\tilde{\boldsymbol{c}}_i\|$, $\ell_{\max} = \max_i \|\tilde{\boldsymbol{c}}_i\|$ of C. This can be visualized in a profile that spreads along a larger range. By analyzing the profile of C as a straight line, following the Geometric Series Assumption, we roughly expect that

$\ell_{\max} \sim \sigma_{\text{pk}}$ and $\ell_{\min} \sim \sigma_{\text{pk}}^{-1}$, and thus $(\ell_{\max}/\ell_{\min}) \sim \sigma_{\text{pk}}^2$ for their ratio.

By setting $q = p^k \leq \ell_{\max}/\ell_{\min}$ with $k = \lfloor \log_p(\ell_{\max}/\ell_{\min}) \rfloor$ we can have a modulus q up to roughly σ_{pk}^2 as well while creating a good basis D for the ambient space $\mathcal{D}$ where decoding BDD instances is non trivial. The value of $\sigma_{\text{enc}} > 0$ used to sample encryptions, on the other hand, depends on the quality of the basis D. In particular, because of Lemma 8, and for security reasons, we require σ_{enc} to be slightly larger than $\max_i \|\tilde{\boldsymbol{d}}_i\| = \max_i \|\tilde{\boldsymbol{c}}_i\| \sim \sigma_{\text{pk}}$. Note that while σ_{enc} used to sample an encryption error $\boldsymbol{e}$ might be large, what matters is the decryption noise coming from $|\langle \boldsymbol{e}, \boldsymbol{s} \rangle|$ which is roughly of size $\|\boldsymbol{s}\| \cdot \sigma_{\text{enc}} \sim \|\boldsymbol{s}\| \cdot \sigma_{\text{pk}}$, in contrast to the modulus $q \sim \sigma_{\text{pk}}^2$. In particular, after a BGV-multiplication one would obtain a decryption noise of roughly $(\|\boldsymbol{s}\| \cdot \sigma_{\text{pk}})^2$ which needs to be smaller than q. To allow for many multiplications we would therefore preferably have an exceptionally short vector $\boldsymbol{s} \in \mathcal{L}$, or equivalently, start with a lattice such that $\boldsymbol{b}_1 \in \mathcal{L}$ is roughly of norm σ_{pk}^{-1}. Note that this matches the LWE case where after normalizing the volume $\|\boldsymbol{s}\| \sim 1/\sqrt{q} \approx \sigma_{\text{pk}}^{-1}$. One way to obtain this is similar to the instantiation proposed in [DvW22], picking $\mathcal{L} := \frac{1}{\lambda}\mathcal{L}' \oplus \lambda\mathcal{L}'$ for one's favourite lattice $\mathcal{L}'$, where λ allows to arbitrarily decrease its first minimum.

4.7 Further extensions and open questions

We presented a lattice-based perspective of two well-known homomorphic encryption schemes: BGV and GSW. This opens up new possibilities in terms of constructions, hardness assumptions and for identifying what could potentially be improved.

Multiple message packing, better decoding and robustness. We encoded a single message in each ciphertext and used a single short vector $\boldsymbol{s} \in \mathcal{L}$ to decrypt it. The generalization to multiple messages is rather straightforward by going from one short secret vector to $k > 1$ short vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k \in \mathcal{L}$ to pack k messages in a ciphertext at once. One does have to assume some additional modular constraints between the ambient basis D and these vectors, but these can be accounted for in the key generation.

More generally, one could consider a dense sublattice $S \subset \mathcal{L}$ of rank $k \geq 1$, along with an efficient decoding algorithm for S^*. This could significantly increase the unique decoding radius $\frac{1}{2}\lambda_1(qS^*)$ of S^*, and therefore allow for larger errors before the decryption fails. Such a more general approach, along with more complex message encodings, for example using linear codes in $\mathbb{F}_p^k \cong S^*/pS^*$, could also allow for a trade-off between the number of messages that can be packed into a ciphertext versus their robustness against errors.

Better modulus switching and analysis. Besides the usage of better lattices and decoding algorithms in S^*, one could also use better lattices and decoding algorithms for the ambient space in the modulus switching procedure. While the ambient space $\mathbb{Z}^m$ used in LWE already presents a simple decoding algorithm and covering radius, it is not optimal, and better alternatives exist such as the A_n root lattices, or direct sums of the Leech lattice (see [CS13] for many more interesting lattices).

[CS13]: Conway and Sloane (2013), Sphere packings, lattices and groups

Alternative benefits can come from replacing the approximate syndrome decoder with a Discrete Gaussian Sampling procedure. While this might increase the decoding errors slightly it should allow for an easily provable average-case analysis of the error growth. Another idea is to keep track of these *known* decoding errors, to potentially compensate for them in a later modulus switching step.

Sparse errors. On a theoretical side the LWE problem is best studied and understood using a (Discrete) Gaussian error distribution over $\mathcal{D} = \mathbb{Z}^m$. In practice however, the error distributions are often heuristically chosen and much narrower, e.g. opting for sparse binary or ternary errors, to decrease the size of the errors. Unfortunately, the specific sparse structure leads to an increase risk from hybrid attacks [EJK20; Nol+24]. One way to avoid this problem is to use Discrete Gaussian sampling on an ambient lattice with a sufficiently small *smoothing parameter* (see [EWY23; BLL25] for examples of such lattices). Doing so, the sampled errors $\boldsymbol{e} \in \mathcal{D}$ can still be close to the minimum distance $\lambda_1(\mathcal{D})$, while provably maintaining a sufficiently high entropy to make hybrid attacks ineffective.

[EWY23]: Espitau, Wallet, and Yu, *'On Gaussian sampling, smoothing parameter and application to signatures'* . In: ASIACRYPT '23

[BLL25]: Bollauf, Lie, and Ling, *'On Gaussian sampling for q-ary lattices and linear codes with Lee weight'* . In: CRYPTO '25

Structured versions. We presented the schemes for plain lattices to focus on the geometric intuition and ideas. This

makes the schemes rather impractical in their current form. One could significantly improve this by using structured *module lattices*, and this generalization follows quite immediately by plugging in such a lattice. This also directly explains why RLWE-based schemes allow to pack multiple messages, as for example in a module lattice over a cyclotomic number field one could multiply the short vector $\boldsymbol{s}$ by a root of unity to obtain another short vector, leading to d short vectors in $\mathcal{L}$ where d is the degree of the number field. So while algebraic structure allows for message packing, it is not a necessity and merely follows from the geometric change of the involved lattice.

Another significant benefit of introducing module structure is that it generally decreases the dimension of $\mathcal{D}/q\mathcal{L}^*$ by a factor d, and therefore the dimension after a BGV multiplication by a factor d^2. The extreme case would be the NTRU lattice where $\mathcal{D}/q\mathcal{L}^*$ has dimension 1, and thus still $1^2 = 1$ after a multiplication. Unfortunately, this property also directly lead to the overstretched attacks on FHE-based NTRU schemes [ABD16; KF17; DW21]. This raises the question: can we obtain such a 1-dimensional ciphertext space in another way while avoiding the overstretched attack, or are these inherently connected?

Identifying improvements and limits. In this work we have coupled the performance of HE operations directly to geometric and algorithmic lattice properties. For example the modulus switching to the covering radius, the error sampling to Discrete Gaussian Sampling and therefore indirectly to the smoothing parameter, and the error ceiling to the minimal distance of the dual S^* of the secret dense sublattice. This coupling, along with the knowledge of bounds on how good these properties can get, allows us to directly see where LWE-based schemes are close to optimal, and in which parts they could be improved and for how much. In particular, these properties show where LWE-based schemes and their simple short-dual-vector decoding approach reach their limits.

Key switching and bootstrapping. We did not discuss the bootstrapping procedure, although although standard techniques should easily be applied to our schemes, since the decryption circuit is of the sample simple form as in LWE. The bootstrapping and key switching method are mostly making use of the homomorphic properties and the group structure of the ciphertext space, and are not really specific to lattices. This opens the question: can key switching and/or bootstrapping

be performed differently, considering specific families of lattices or specific lattice operations? For example, the main goal of key switching is to move from a large dimensional lattice to a lower dimensional one. For lattices such an operation is naturally associated with projections, and in general, BDD instances can still be BDD instances after an appropriate projection. Could we therefore replace or rephrase key switching as some lattice projection?

Acknowledgements

A large part of the work presented in this chapter was carried out while W. van Woerden was employed, and while L. Rovida was hosted by IMB at the University of Bordeaux. There, they were both supported by the CHARM ANR-NSF grant (ANR-21-CE94-0003).

4.A Appendix: security proofs

4.A.1 IND-CPA security proof

In this section we will consider the IND-CPA security proof of our Lattice-BGV and Lattice-GSW HE schemes based on Assumption 1. For simplicity we consider the setting that the evaluation or bootstrapping keys are encrypted under a fresh keypair as to prevent any circularity assumptions. As a result we can simply consider the IND-CPA security of the underlying PKE schemes.

Theorem 10

The Lattice-BGV and Lattice-GSW schemes are IND-CPA secure under Assumption 1.

Proof. Both schemes and Assumption 1 share the same key generation procedure *Keygen* defined in Definition 62. Consider one scheme and let $pk, sk = (B, D), s \leftarrow Keygen(1^\lambda)$, $C := B^{-t}, \mathcal{D} := \mathcal{L}(D)$ and consider the parameters q, t, m and σ_{enc} from the scheme. Let $\mathcal{A}(pk)$ be an adversary breaking the IND-CPA security of the scheme. Consider an instance of the $\Delta SDP_{C,D,\sigma_{\text{enc}}}$ problem given sufficiently many syndromes $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\text{poly}(m)} \in \mathcal{P}(qC) \cap \mathcal{D}$. We use $\mathcal{A}$ to break the $\Delta SDP_{C,D,\sigma_{\text{enc}}}$ problem. Here the proof slightly deviates for the two schemes.

Lattice-BGV. We consider a single syndrome $\boldsymbol{c}_1$, query the adversary $\mathcal{A}$ for a pair of messages (m_0, m_1) and sample a uniform bit $b \leftarrow \{0,1\}$. We return the challenge $\boldsymbol{c}^* := t \cdot \boldsymbol{c}_i + D \cdot (m_b, 0, \ldots, 0) \bmod \mathcal{P}(qC)$, record the response $b' \leftarrow \mathcal{A}$ from the adversary, and return $b + b' \bmod 2$ to the $\Delta SDP_{C,D,\sigma_{\text{enc}}}$ problem. Note that if $\boldsymbol{c}_1$ is sampled from $\mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ then $\boldsymbol{c}^*$ follows the same distribution as a regular Lattice-BGV encryption of m_b and therefore the adversary should be biased towards the correct guess $b' = b$ and thus towards $b + b' = 0 \bmod 2$. Otherwise $\boldsymbol{c}_1$ is uniform, and using that $\gcd(q,t) = 1$ for Lattice-BGV we obtain that $t \cdot \boldsymbol{c}_i + D \cdot (m_b, 0, \ldots, 0)$ is still uniform and statistically independent from b, and thus $b + b' \bmod 2$ is 1 half of the time. We thus solve the distinguishing problem $\Delta SDP_{C,D,\sigma_{enc}}$ with essentially the same complexity and advantage as $\mathcal{A}$.

Lattice-GSW. We proceed similarly but we now consider $\log_p(|\mathcal{D}/q\mathcal{L}^*|) \leq m \log_p(q)$ syndromes (where $q = p^k$). Note that if the syndromes have an error sampled from $\mathcal{G}_{\mathcal{D},\sigma_{\text{enc}}}$ we can add μDG to it modulo $\mathcal{P}(qC)$ for the appropriate gadget matrix G to exactly obtain a Lattice-GSW encryption of μ with the correct distribution. From here the argument is identical to the Lattice-BGV case. □

4.A.2 NTRU and LWE-based instantiations

To indicate that the key generation (Definition 62), the security assumption (Assumption 1) and therefore also the IND-CPA security proof are proper generalisations of the original LWE-based BGV and GSW schemes we shortly discuss their instantiation. Recall that one can instantiate the key generation procedure with LWE by creating a lattice $\mathcal{L} := \mathcal{L}_q(A)$ with a short trapdoor vector $\boldsymbol{s} \in \mathcal{L}_q(A)$. Implicitly this creates a (public basis of) the q-dual lattice $\mathcal{L}_q^\perp(A)$ along with an identity basis I_m of the ambient space $\mathcal{D} := \mathbb{Z}^m$.

As explained before the security proof proceeds in two steps. Firstly, by the LWE assumption (the public basis of) $\mathcal{L}^\perp(A)$ is indistinguishable from a random q-ary lattice. After replacing $\mathcal{L}^\perp(A)$ with such a random q-ary lattice in the security game, we see that Assumption 1 becomes precisely the (syndrome version of) the LWE assumption, and the remainder of the proof proceeds as in Theorem 10.

Note that one could similarly also replace the key generation procedure with an NTRU-based assumption. As we do not

consider module lattices in this work we will simply consider the matrix-NTRU assumption. With matrix-NTRU one can generate a lattice $\mathcal{L}$ with a trapdoor dense sublattice $S \subset \mathcal{L}$, which in particular contains a short trapdoor vector $s \in S \subset \mathcal{L}$. Under a distinguish matrix-NTRU assumption this lattice is again indistinguishable from a random q-ary lattice. In this setting Assumption 1 thus again becomes the LWE assumption, and the proof proceeds as before. One can thus also instantiate our lattice framework with a combination of the NTRU and LWE assumptions, which is precisely the strategy that is followed by modern FHE schemes like FINAL [Bon+22].

[Bon+22]: Bonte, Iliashenko, J. Park, Pereira, et al., *'FINAL: faster FHE instantiated with NTRU and LWE'* . In: ASIACRYPT '22

4.A.3 Security proof under LIP instantiation

In Section 4.6 we considered an instantiation of our Lattice-BGV scheme based on the Lattice Isomorphism Problem. Here we give some more details how to obtain an IND-CPA security proof in this specific setting. We can consider two general approaches for this. First, one can consider the approach considered in the work by Ducas and van Woerden [DvW22] to prove the IND-CPA security of a KEM purely based on ΔLIP (distinguish-LIP) between two lattices. When combined with further tightness improvements in [vW24] one can create a relatively efficient IND-CPA security proof based on this single assumption. The disadvantage is that there is still some tightness loss in the security reduction, and that the scheme needs to be adjusted slightly to instantiate this particular security proof. This is the approach that is taken in [BMM25] (without the optimizations of [vW24]).

[DvW22]: Ducas and van Woerden, *'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'* . In: EUROCRYPT '22

[vW24]: van Woerden, *'Dense and smooth lattices in any genus'* . In: ASIACRYPT '24

[BMM25]: Branco, Malavolta, and Maradni (2025), Fully-Homomorphic Encryption from Lattice Isomorphism (Cryptology ePrint Archive, Paper 2025/993)

A potentially more natural approach, following the ideas of [vW24] and matching our framework including Assumption 1, is to consider a two step approach in the security proof just as for LWE (or as for the NTRU+LWE combination). This would also require two assumptions instead of a single one. The first assumption is precisely the distinguish variant ΔLIP between the lattice $\mathcal{L}$ used in our scheme and a random lattice $\mathcal{L}'$ in the same genus of $\mathcal{L}$. As the genus is currently the best-known efficiently computable invariant for LIP, the state-of-the-art cryptanalysis states that it hard to solve ΔLIP between $\mathcal{L}$ and $\mathcal{L}'$ in the same genus. This assumption allows to replace $\mathcal{L}$ with the random lattice $\mathcal{L}'$ in the security proof. The second assumption, matching Assumption 1 in this setting, would (informally) be that it is hard to solve a syndrome decoding problem in a random lattice $\mathcal{L}'$ in the genus of $\mathcal{L}$. This second assumption is a very light assumption as [vW24]

indicates that random lattices in such a fixed genus behave just like general random lattices, which are generally seen as the hardest instances for decoding problems.

Informally we thus obtain a LIP-based HE based on the following two assumptions. For more details on distinguish-LIP, the genus, and the precise notion of randomness, we refer to [DvW22; vW24].

Assumption 2 (*Distinguish* LIP *(informal)*)

Given $\mathcal{L}$ and a random lattice $\mathcal{L}' \leftarrow \mathit{Genus}(\mathcal{L})$ following the standard Haar measure on the space of lattices restricted to $\mathit{Genus}(\mathcal{L})$, it is hard to distinguish between $\mathcal{L}$ and $\mathcal{L}'$.

Assumption 3 (*Decoding in random lattice (informal)*)

Let $\mathcal{L}$ be a lattice and consider a random lattice $\mathcal{L}' \leftarrow \mathit{Genus}(\mathcal{L})$. Then solving the syndrome decoding problem in $\mathcal{L}'$ is hard.

To formalize these assumptions further one has to consider the distribution of the representation of $\mathcal{L}$ and $\mathcal{L}'$ (see [DvW22]), and to consider the precise error parameter σ_{enc}.

Algorithms

Classification of encrypted images

5

This chapter is an extended/revised version of 'Encrypted Image Classification with Low Memory Footprint' *co-authored with A. Leporati, published in International Journal of Neural Systems*

Abstract: Classifying images has become a straightforward and accessible task, thanks to the advent of deep neural networks. Nevertheless, not much attention is given to the privacy concerns associated with sensitive data contained in images. In this study, we propose a solution to this issue by exploring an intersection between Machine learning and cryptography. In particular, Fully homomorphic encryption (FHE) emerges as a promising solution, as it enables computations to be performed on encrypted data. We, therefore, propose a Residual network implementation based on FHE which allows the classification of encrypted images, ensuring that only the user can see the result. We suggest a circuit which reduces the memory requirements by more than 85% compared to the most recent works, while maintaining a high level of accuracy and a short computational time. We implement the circuit using the well-known CKKS scheme, which enables approximate encrypted computations. We evaluate the results from three perspectives: memory requirements, computational time and calculations precision. We demonstrate that it is possible to evaluate an encrypted ResNet20 in less than five minutes on a laptop using approximately 15GB of memory, achieving an accuracy of 91.67% on the CIFAR-10 dataset, which is almost equivalent to the accuracy of the plain model (92.60%).

5.1 Introduction

In recent years, deep neural networks have demonstrated impressive capabilities across various tasks, and they are progressively becoming a fundamental block in delivering many online services, especially in the image recognition task [LBH15].

Nevertheless, not enough attention is given to user privacy when these models are deployed as *Machine Learning as a Service* (MLaaS). In order to run inferences, service providers

[LBH15]: LeCun, Bengio, and G. Hinton (2015), *'Deep learning'* . In: Nature

need to access plain user data, and this creates issues that are not easy to deal with. Today, data is protected by laws, like the General Data Protection Regulation (GDPR) [EC16a] in Europe, but there is still a urgent need to have strong cryptographic security for data. We propose a solution by exploring an intersection between Machine learning and cryptography. In particular, we implement a well-known Convolutional neural network (CNN) called Residual neural network (ResNet), using a Homomorphic encryption scheme that allows for operations to be performed on encrypted data.

[EC16a]: European Parliament and Council of the European Union (2016), Regulation (EU) 2016/679 of the European Parliament and of the Council

5.1.1 Convolutional neural networks

CNNs have emerged as a powerful class of deep learning models designed to handle data with a grid-like structure, especially images. They were first introduced in 1989 by Le Cun *et al*. [Le +90] as a method to recognize handwritten digits.

The main idea behind these networks is to capture spatial dependencies, local patterns and features from data, making them highly effective for the image classification task. In particular, in this chapter, we present an FHE-based version of a Residual neural network called ResNet20. This family of CNNs is based on Residual connections (also known as Skip connections), a shortcut in a Neural network that allows the gradient to bypass one or more layers during training. It adds the original input of a layer to its output, creating a Residual block and allowing for the network to learn and optimize the parameters effectively. Residual blocks indeed prevent the issue of the vanishing gradient problem.

[Le +90]: Le Cun, Boser, Denker, Howard, et al. (1990), *'Handwritten digit recognition with a back-propagation network'* . In: Advances in Neural Information Processing Systems 2

5.1.2 Related works

Inference from encrypted images based on CNNs has been object of study since few years. The first generation of CNNs based on HE, called *HE-friendly*, is characterized by networks based on polynomial – in most cases linear or square – activation functions, as a consequence of the limited set of arithmetic operations in HE (i.e. only additions and multiplications).

One of the first successful attempts was CryptoNets, by Gilad-Bachrach *et al.* in 2016 [Gil+16], which was able to obtain a 99% of accuracy on the MNIST dataset using a simple CNN. In 2021, Al Badawi et al. [Al +21] proposed a CNN based on

[Gil+16]: Gilad-Bachrach, Dowlin, Laine, Lauter, et al., *'CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy'* . In: PMLR '16

[Al +21]: Al Badawi, Jin, Lin, Mun, et al. (2021), *'Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data With GPUs'* . In: IEEE Transactions on Emerging Topics in Computing

RNS-CKKS accelerated by GPU, achieving 77.55% accuracy on the CIFAR-10 dataset, firstly introduced in [KH09].

[KH09]: Krizhevsky and G. Hinton (2009), Learning multiple layers of features from tiny images

The main issue about HE-friendly networks is that they do not implement non-linear activation functions, and this implies lower performance on more difficult tasks. Nevertheless, recent developments on FHE schemes, especially on bootstrapping techniques, are enabling practical evaluations of deeper and more complex circuits, meaning that is it possible to use existent and pre-trained networks, instead of building and training new networks specifically for HE computations. The problem of non-linear functions is tackled using polynomial approximations.

For this second generation of networks, we find two approaches: *High-throughput networks* and *Low-latency networks*. The former work on many images simultaneously, minimizing the amortized inference runtime. This approach, though, work well when dealing with big chunks of images, and not with single images. In this work we will discuss and propose a Low-latency network.

In particular, these networks are tackled in two different ways. The first one is to use the TFHE scheme. A recent work by Benamira *et al.*, in 2023 [Ben+23a], presents an architecture that evaluates an encrypted image in approximately nine minutes, achieving a performance of 74.1% on the CIFAR-10 dataset, requiring less than 1GB of memory.

[Ben+23a]: Benamira, Guérand, Peyrin, and Saha (2023), TT-TFHE: a Torus Fully Homomorphic Encryption-Friendly Neural Network Architecture

The second approach is to use the CKKS scheme. The following works are the ones that will be used as reference to compare the different approaches to Convolutional Layers. Lee et al. [Lee+22b] proposed in 2022 a framework able to obtain a high level of accuracy (91.31% on CIFAR-10) in half an hour of computations. Their convolution approach follows the work of Juvekar *et al.* [JVC18a], and we will refer to it as *Vector encoding*. Kim and Guyot [KG23b] introduced, one year later, an open-source framework that is able to achieve a classification with a level of accuracy of 92.04% on the CIFAR-10 dataset in approximately six minutes. Their key point is in the encoding process, although this approach is characterized by a high memory usage (100GB). We refer to this approach as *Coefficient encoding*. The following are the two main HE-based convolution approaches:

[Lee+22b]: E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, et al., *'Low-Complexity Deep Convolutional Neural Networks on Fully Homomorphic Encryption Using Multiplexed Parallel Convolutions'* . In: PMLR '22

[JVC18a]: Juvekar, Vaikuntanathan, and Chandrakasan, *'GAZELLE: A Low Latency Framework for Secure Neural Network Inference'* . In: USENIX '18

[KG23b]: D. Kim and Guyot (2023), *'Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption'* . In: IEEE Transactions on Information Forensics and Security

- Vector encoding [JVC18a; Lee+22b]: this approach requires k^2 (where k is the size of the kernel) ciphertexts representing different rotations of the input image. Each ciphertext is multiplied with the corresponding kernel

elements for each channel, leading to $k^2 \cdot c$ multiplications, where c is the number of channels. This approach requires $k^2 + c$ rotations and rotation keys.

- Coefficient encoding [KG23b]: this approach encodes an input vector directly as polynomial coefficients (i.e., *coeff* encoding, without application of the σ^{-1} encoding procedure), and convolutions are performed in a single step by multiplying two polynomials. Nevertheless, many rotations keys are required to extract the resulting coefficients, and this implies a high memory usage.

Our contribution. In our proposal, we utilize the CKKS scheme to implement a trained ResNet20. Differently from the most recent works [Lee+22b; KG23b], we minimize the memory requirements. De Castro *et al.* [Cas+21] showed that memory is currently the main bottleneck to be addressed in FHE circuits, although most of the works do not consider its impact when building FHE solutions. We thus introduce an efficient way to perform convolutions, namely *Optimized vector encoding*, requiring only five rotation keys. This approach allows to save a lot of memory, and complexity is not increased with respect to the standard Vector encoding procedure. Our work compares to current literature as shown in Figure 69 and in Table 6.

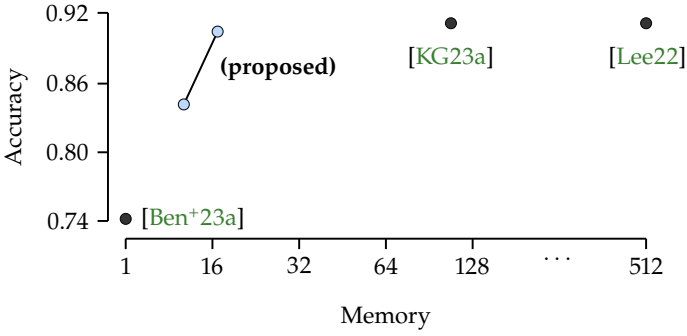


Figure 69: Accuracy and memory usage of existent literature work and our proposal

Security model. The architecture of our work is composed of two actors: a honest-but-curious server, which offers a classification service, and a client, which asks for a classification. The client generates a set of keys and shares the public keys to the server. We propose a HE-based circuit that the server is able to evaluate on encrypted data, so that the client only, which owns the secret key, is able to decrypt the output result. In practice, this means that the server operates on an image which is

Table 6: Comparison of the most recent solutions for Low-Latency CIFAR-10 HE-based CNN inference. * denotes that the authors did not provide information about the memory usage, we thus assume the RAM capacity of their machine

Proposal	Year	Scheme	Accuracy	Runtime	Threads	Memory
[Lee22]	2022	CKKS	90.67%	10602s	64	512 GB*
[Lee+22b]	2022	CKKS	91.31%	2271s	1	512 GB*
[Ben+23a]	2023	TFHE	74.10%	570s	4	0.8 GB
[KG23b]	2023	CKKS	92.04%	255s	1	100 GB
(proposed)	2024	CKKS	91.53%	260s	1	**15.1 GB**

encrypted. As a consequence, neither an external attacker or the server itself are able to see the content of the image sent by the client, and thus the user privacy is verified.

5.1.3 Preliminaries

We quickly re-introduce and review some essential preliminaries concerning FHE and CKKS, although we refer to Section 3.3 for further details.

Mathematical fundamentals. The ring of polynomials with integer coefficients is represented as $\mathbb{Z}[X]$, and we define $\mathcal{R} := \mathbb{Z}[X]/(X^n + 1)$ as the ring of polynomials modulo X^n+1, where n is a power of two. The original CKKS [CKKS17] scheme takes as input a so-called *cleartext* $v \in \mathbb{C}^{n/2}$, and it first encodes it into a polynomial in $\mathcal{R}$. In Section 3.3 we saw how a cleartext in $\mathbb{C}^{n/2}$ is encoded in a polynomial in $\mathbb{Z}[X](X^n + 1)$, and we call this encoding *Slots* encoding. On the other hand, for some applications, it might be convenient to encode a vector $\boldsymbol{v} \in \mathbb{Z}_q^n$ in a polynomial, namely *Coeff* encoding. SIMD computations are harder to obtain under this setup, but convolutions can be computed more easily as no slot-wise multiplications are performed. It is possible to encode it in two different ways:

- *Coeff*: the vector v is encoded directly as polynomial coefficients. It allows to pack more values (n) in a single ciphertext, but the product of two *coeff* polynomials in $\mathcal{R}$ does not result in point-wise multiplication between coefficients.
- *Slots*: the vector v is subject to a canonical embedding $\sigma^{-1} : \mathbb{C}^{n/2} \to \mathbb{Z}[X]/(X^N + 1)$ which halves the number of available slots in a ciphertext, but negacyclic convolutions in $\mathcal{R}$ will result in a slot-wise product in the original input

space $\mathbb{C}^{n/2}$. In other words, this encoding process enables point-wise (or slot-wise) multiplications between ciphertexts.

After the encoding, coefficients are scaled by a factor Δ that controls the precision of computations. Larger values of Δ allow to save more digits of the encoded plaintext.

Remark 9

The scheme works with polynomials in $\mathbb{Z}[X]/(X^n+1)$, with n being a power of two, because they allow for fast computations using the NTT, which allows one to perform multiplications with complexity $O(n\log(n))$ instead of the basic modular polynomial multiplication, which has a complexity $O(n^2)$.

In particular, we use the RNS variant of the CKKS scheme. It works on DCRT polynomials. A DCRT polynomial is a large polynomial factorized in smaller ones using CRT and transformed in the NTT domain, for performance reasons, refer to Section 3.1.

After the encoding, the plaintext polynomial is encrypted using the ring-LWE problem. Informally, we consider as the public key a RLWE sample $(\boldsymbol{a},\boldsymbol{b}) \in (\mathbb{Z}_q[X]/(X^n+1))^2$ such that $\boldsymbol{a} \in \mathcal{R}_q$ is uniformly sampled and $\boldsymbol{b} = \boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e}$ for a small $\boldsymbol{e}$. Refer to Section 3.3 for more details about the used distributions. The ciphertext $\boldsymbol{c}$ encrypting the input polynomial $m(X)$ is thus obtained as follows:

$$\mathbf{c} = \boldsymbol{u}\cdot(\boldsymbol{a},\boldsymbol{b}) + (\boldsymbol{m}+\boldsymbol{e}_1,\boldsymbol{e}_2) \in (\mathbb{Z}_q[X]/(X^n+1))^2,$$

with $\boldsymbol{u} := \mathcal{ZO}(0.5)$ and $\boldsymbol{e}_1,\boldsymbol{e}_2 \sim \mathcal{DG}(\sigma)$. An important feature of the CKKS cryptosystem is that it is possible to pack multiple values in a single polynomial, leading to a Single Instruction Multiple Data (SIMD) computational paradigm, which unlocks parallel computations.

Our proposal is based on the OpenFHE implementation [Al+22] of the RNS-CKKS [KPP22] scheme, which allows to perform operations on encrypted $\mathbb{R}^{n/2}$ vectors. Remark that computations are approximate, meaning that:

$$\boldsymbol{v} = \mathrm{Dec}(\mathrm{Enc}(\boldsymbol{v})) + \boldsymbol{\varepsilon}$$

where $\boldsymbol{\varepsilon}$ is a small error vector introduced by different factors (by the rounding in encoding process, by the scaling factor and because of RLWE encryption) which grows when performing operations. The available operations are addition ($\boldsymbol{c}_1+\boldsymbol{c}_2$), multiplication ($\boldsymbol{c}_1\cdot\boldsymbol{c}_2$) and rotation $\mathrm{Rot}_i(\boldsymbol{c})$ (when $i>0$, coefficients are rotated to the left, to the right otherwise).

[Al+22]: Al Badawi, Bates, Bergamaschi, Cousins, et al., *'OpenFHE: Open-Source Fully Homomorphic Encryption Library'* . In: WAHC '22

[KPP22]: A. Kim, Papadimitriou, and Polyakov, *'Approximate Homomorphic Encryption with Reduced Approximation Error'* . In: CT-RSA '22

In case of multiple rotations on the same ciphertext, the hoisted automorphisms technique [HS14] FastRot$_i(c)$ is used. Notice that additions and multiplications can also be evaluated between ciphertexts and plaintexts (with a more convenient error growth, as plaintexts do not contain large errors).

[HS14]: Halevi and Shoup, *'Algorithms in HElib'* . In: CRYPTO '14

Chebyshev polynomials. Our circuit uses an algorithm to efficiently evaluate this set of polynomials using a modified version of the Paterson-Stockmeyer [PS73] algorithm introduced by Chen, Chillotti and Song [CCS19] (refer also to Section 3.3.1). In our proposal, we used Chebyshev polynomials to approximate the ReLU activation function and the modular reduction required by the bootstrapping procedure.

[PS73]: Paterson and Stockmeyer (1973), *'On the number of nonscalar multiplications necessary to evaluate polynomials'* . In: SIAM Journal on Computing

[CCS19]: H. Chen, Chillotti, and Song, *'Improved Bootstrapping for Approximate Homomorphic Encryption'* . In: EUROCRYPT '19

5.2 Circuit overview

We now present the FHE-based circuit that implements ResNet20. Before that, we give a brief description of the network layers.

Convolutional layer. A convolutional layer is a key component of a Convolutional neural network (CNN) [Le +90] that applies a set of learnable filters to an input matrix. Each filter performs a convolution operation by sliding across the input and computing dot products between the filter weights and local regions of the input. The output matrices, called feature maps, capture spatial patterns such as edges, textures, or shapes. Mathematically the output is Y, a set of n matrices indexed by $Y^{(k)}$, where each $y_{i,j}^{(k)}$ is:

[Le +90]: Le Cun, Boser, Denker, Howard, et al. (1990), *'Handwritten digit recognition with a back-propagation network'* . In: Advances in Neural Information Processing Systems 2

$$y_{i,j}^{(k)} = \sum_{m=1}^{w} \sum_{n=1}^{w} x_{i+m,\, j+n}\, w_{m,n}^{(k)} + b^{(k)}, \tag{5.1}$$

where X is the input matrix (and $x_{i,j}$ is the element at position i, j of X), $W^{(k)}$ is the k-th filter (kernel) matrix of size $w \times w$, $b^{(k)}$ is a scalar bias term, and $Y^{(k)}$ is the output feature map matrix for the k-th filter. Observe that k depends on how many output channels are in defined in the layer.

Batch normalization. Batch normalization (BN) [IS15] stabilizes and accelerates training by normalizing activations. For a mini-batch of size m, let $\boldsymbol{x}_i$ be the vector of activations for a single feature (or channel) in sample i. The BN steps are:

[IS15]: Ioffe and Szegedy, *'Batch normalization: accelerating deep network training by reducing internal covariate shift'* . In: ICML'15

1. Compute the batch mean:

$$\mu_B := \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$$

2. Compute the batch variance:

$$\sigma_B^2 := \frac{1}{m}\sum_{i=1}^{m} (\boldsymbol{x}_i - \mu_B)^2$$

3. Normalize each activation:

$$\hat{\boldsymbol{x}}_i := \frac{\boldsymbol{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

4. Scale and shift (learnable parameters):

$$\boldsymbol{y}_i := \gamma \hat{\boldsymbol{x}}_i + \beta$$

where γ and β are scalar learnable parameters per channel, $\varepsilon > 0$ is a small constant for numerical stability and $\boldsymbol{y}_i$ is the output vector of normalized and rescaled activations for sample i. Batch normalization is applied per feature channel, typically after the linear or convolution operation and before the activation function. During the inference, we use constant terms for γ and β.

The FHE circuit. We are interested to study the multiplicative depth of the circuit, which we distribute as follows:

- 1 level consumed for the Convolutional layer and the Batch normalization, combined.
- 5, 6 or 7 levels, depending on the desired precision, consumed for the evaluation of the Chebyshev polynomial approximating ReLU.

After that, we refresh the level of the ciphertext via the bootstrapping procedure (refer to Section 5 of Boussat *et al*. [Bou+21] for a detailed exploration):

- 1 level consumed after the `ModRaise` procedure.
- 4 levels consumed by the `CoeffToSlot` procedure, i.e. the homomorphic version of the encoding.
- 5 levels consumed by the Chebyshev approximation of $g_0(x)$, that is the base case of the modular reduction iterative procedure (with r equal to the number of

[Bou+21]: Boussat, Mouchet, J. Troncoso-Pastoriza, and Habaux, '*Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys*' . In: EUROCRYPT '21

iterations):

$$g_0(x) = \frac{1}{\sqrt[2^r]{2\pi}} \cos\left(\frac{2\pi}{2^r}\left(x - \frac{1}{4}\right)\right)$$

- 3 levels consumed by the double-angle iterations used to conclude the evaluation of the homomorphic modular reduction:

$$g_{i+1}(x) = 2\, g_i^2 - \left(\frac{1}{\sqrt[2^r]{2\pi}}\right) \text{ with } 0 \leq i < r$$

- 4 levels consumed by the `SlotToCoeff` procedure, i.e. the homomorphic version of the decoding.

So, in total, we require ciphertexts that allow for roughly 25 levels (8 levels for the homomorphic operations and 17 levels to refresh the level of the ciphertext).

5.2.1 Residual Networks Structure

The implemented network, ResNet20 is composed of 2+3 big layers, constructed as shown in Figure 70.

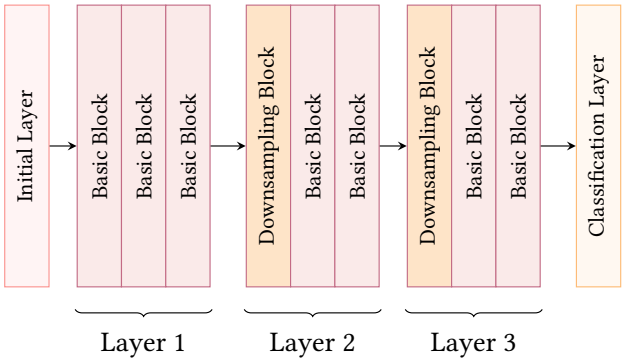


Figure 70: High-level structure of the ResNet20 network

The layers are defined as follows

- Initial Layer: takes as input a RGB image I represented as a tensor $\mathbb{R}^{3\times32\times32}$ and evaluates a Convolutional layer, a Batch normalization and the ReLU function; it returns a $\mathbb{R}^{16\times32\times32}$ tensor.

- Layer 1: this layer is composed of three basic block, each of them built as shown in Figure 71. Each block works with $16 \times 32 \times 32$ input tensors, Convolution Layers have a padding of $\{1, 1\}$, a kernel width $k = 3$, and a stride of $\{1, 1\}$ The output is a tensor $\in \mathbb{R}^{16\times32\times32}$.

► Layer 2: this layer performs a downsampling on the input image by performing the first convolution with a stride of $\{2, 2\}$. This halves the width of the $n_{out} = 32$ channels. As a result, the output of the downsampling block is a tensor $\in \mathbb{R}^{32\times16\times16}$. Then, two basic blocks follow.

► Layer 3: as before, there is another downsampling at the beginning of this layer. In this case $n_{out} = 64$, so the output tensor $\in \mathbb{R}^{64\times8\times8}$. After that, two basic blocks complete the layer.

► Classification Layer: an Average pooling layer is used to extract features from each channel. This is followed by a fully connected layer with 10 output neurons, each corresponding to a specific class. The classification decision is made by selecting the neuron with the highest output score, and it is done by the final user. Namely, the output of the encrypted circuit is a vector containing the 10 values.

The Basis block, present in the three central layers, has the structure presented in Figure 71. The size of the ciphertexts

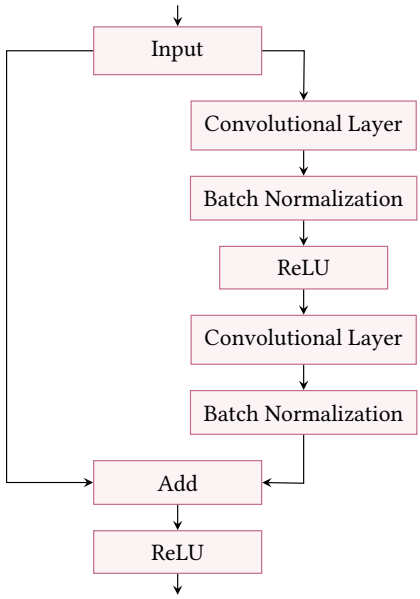


Figure 71: Structure of a basic block in ResNet20

will change through the evaluation of the circuit: at Layer 1, ciphertexts have $s = 16384$ slots, then s is reduced to $s = 8192$, and so on (see Table 7). This allows to perform computations on smaller ciphertexts, which are faster.

Layer	Channels (c)	Size (w)	Total values ($c \cdot w^2$)
1	16	32	$16384 = 2^{14}$
2	32	16	$8192 = 2^{13}$
3	64	8	$4096 = 2^{12}$

Table 7: Image shapes in different ResNet20 layers

5.2.2 Optimized Vector Encoding

We propose an optimized version of the Vector Encoding procedure that, in case of odd-sized kernels with $\{1, 1\}$ padding, uses a constant number of five rotation keys. Notice that the number of rotation keys does not depend on the size of the kernel k^2 or on the number of channels c, while in previous implementations [JVC18a] the number of rotation keys was $(k^2 - 1) + c$. This allows one to drastically reduce the memory footprint of the rotation keys.

[JVC18a]: Juvekar, Vaikuntanathan, and Chandrakasan, *'GAZELLE: A Low Latency Framework for Secure Neural Network Inference'* . In: USENIX '18

Incorporating the Batch normalization. The Optimized Vector Encoding procedure allows to evaluate both the Convolutional Layer and the following Batch normalization layer at the same time. When a trained ResNet is evaluated, all the Batch normalization layers use precomputed values of mean and variance, making it, de facto, an affine transformation. The original formula of the Batch normalization is the following:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}(x) + \varepsilon}} \cdot \gamma + \beta$$

Since we consider the inference process, the value of the mean $\mathrm{E}[x]$ and the variance $\mathrm{Var}(x)$ are not computed (instead, constant values are used), and we can thus re-write the equation as follows:

$$\begin{aligned} y &= \frac{x - E}{V} \cdot \gamma + \beta \\ y &= \frac{\gamma}{V} x - \frac{E\gamma}{V} + \beta \\ y &= Ax + b \quad \text{with } A = \frac{\gamma}{V},\ b = \frac{E\gamma}{V} + \beta \end{aligned}$$

Now, since Convolutional layers can be seen as affine transformations, they can be combined with Batch Normalization, consuming only one multiplicative depth.

Applying the kernel. We introduce the kernel application along with an example, in particular we want to replicate the

convolution shown in Figure 72. Notice that all the convolutions (excluding the ones performed in downsampling) are performed with $\{1, 1\}$ padding. To begin with, the application

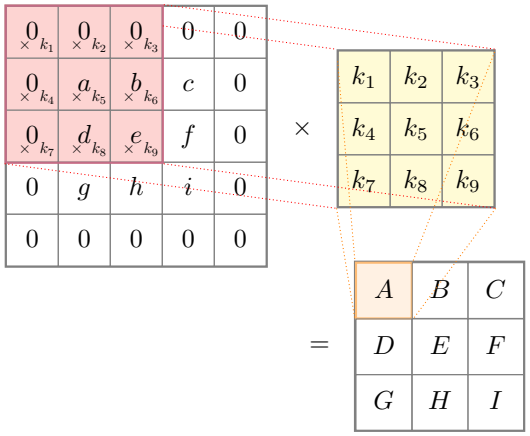


Figure 72: Convolution on a width $w = 3$ channel with padding $= \{1, 1\}$

of the kernel to the image requires a ciphertext containing all the input channels next to each other, from the first to the last one. More formally, let $I \in \mathbb{R}^{c \times w \times w}$ be the input feature map with c channels of dimension $w \times w$. The slots of the ciphertext $\boldsymbol{c}$ encrypting I are such that

$$\boldsymbol{c}_{(w^2 \cdot i)+(w \cdot j)+z} := I_{i,j,z}$$

where $0 \leq i < c$ indexes the channel and $0 \leq j, z < w$ the spatial position. Notice that the values of c and w change during the evaluation of the circuit depending on the considered ResNet20 layer, as shown in Table 7. The input channels are therefore reshaped as shown in Figure 73. The evaluation of

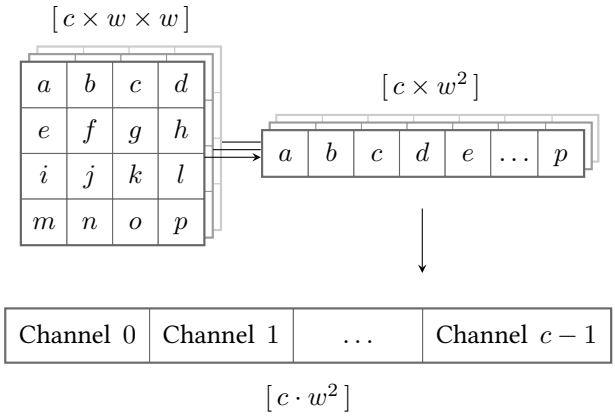


Figure 73: The reshaping process applied before the encryption

the convolution consists of three steps.

1. Preparing the rotations. We start by considering the following:

Lemma 29

All possible rotations of a ciphertext can be obtained with a single rotation key, given an index that is a generator for the ring $\mathbb{Z}_{n/2}$.

Proof. By contradiction, if there is a rotation indexed by $i \in \mathbb{Z}_{n/2}$ that can not be obtained with a generator g of $\mathbb{Z}_{n/2}$, it means that there exists no k such that $g^k = i$, which is impossible since g is a generator of $\mathbb{Z}_{n/2}$. □

Since the dimension of the ring is always a power of two, and its generators are the residues coprime to n, we also have, more specifically:

Corollary 8

Any odd number can be used to obtain any rotation of a ciphertext $\boldsymbol{c}$.

For instance, we can obtain any rotation r of a ciphertext $\boldsymbol{c}$ by applying r times $\text{Rot}_1(\boldsymbol{c})$, and this only requires one rotation key, or in general any odd number. Of course this approach is not suggested since it is extremely time-consuming. By representing each possible rotation as a binary number, one could think of evaluating any rotation using only $\log(n)$ rotation keys, one for each power of two. For instance, to rotate by 19 positions, simply rotate by 16+2+1. This method is optimal in theory, but ad-hoc optimizations work better. Instead, we propose a hybrid approach by finding *possible combinations* of rotation indexes.

Recall our objective, that is to create nine (w^2) ciphertexts which align the nine required values for each filter application. Considering the example in Figure 72, we want to obtain all the required rotations like is shown in Figure 74, where the first column contains the elements needed for the first convolution, the second column contains the elements needed for the second convolution, and so on.

In our example, we would need eight keys for the following indexes: $\{-6, -5, -4, -1, +1, +4, +5, +6\}$, but this set can be reduced to $\{1, -5, +5, +1\}$ by noticing that some of the elements can be written in terms of these four (e.g., $6 = 5 + 1$). Moreover, this method does not require more rotations, since we simply compose existing rotations.

In general, the kernel operates on slices of the feature map that are stored in different rows. Considering that the ciphertext

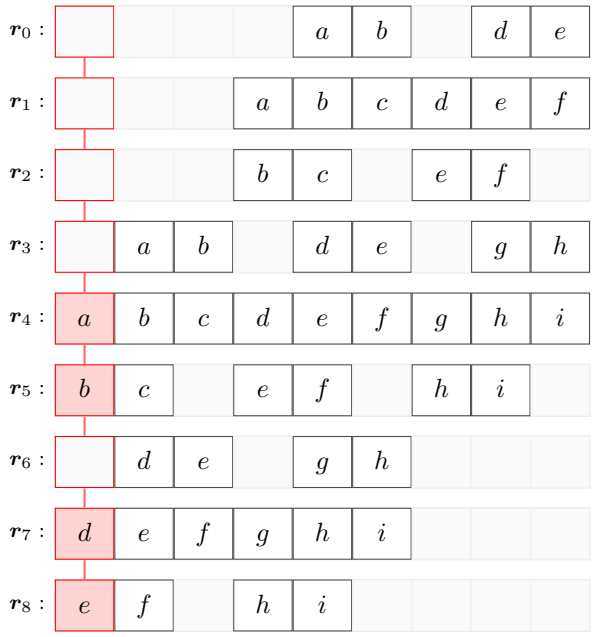


Figure 74: Rotations required for the application of a filter on the input feature map presented in Figure 72. The red column represents the elements used in the first filter application

encodes a bi-dimensional feature map in row-major order, we need keys to rotate "horizontally" inside the slice and "vertically" to rotate into another slice.

Given the feature map size w, and the padded size $w' = w + 2$, we create keys to rotate horizontally for indexes 1 and -1 (right and left), and for indexes w' and $-w'$ (below and above) in order to rotate vertically. We thus align the nine elements required for the filter application as follows:

- $\boldsymbol{r}_0 := \text{Rot}_{-1}(\boldsymbol{r}_1)$
- $\boldsymbol{r}_1 := \text{FastRot}_{-w}(\boldsymbol{r}_4)$
- $\boldsymbol{r}_2 := \text{Rot}_{+1}(\boldsymbol{r}_1)$
- $\boldsymbol{r}_3 := \text{FastRot}_{-1}(\boldsymbol{r}_4)$
- $\boldsymbol{r}_4 := \boldsymbol{c}$
- $\boldsymbol{r}_5 := \text{FastRot}_{+1}(\boldsymbol{r}_4)$
- $\boldsymbol{r}_6 := \text{Rot}_{-1}(\boldsymbol{r}_7)$
- $\boldsymbol{r}_7 := \text{FastRot}_{w}(\boldsymbol{r}_4)$
- $\boldsymbol{r}_8 := \text{Rot}_{+1}(\boldsymbol{r}_7)$

The general procedure is presented in Algorithm 5. In particular, the first for-loop creates the vertical rotations, rotating by w and $-w$. On the other hand, the second loop creates the other rotations (rotating by -1 and 1), using the previously generated ciphertexts.

```
1  function GenerateRotations (c, w)
2      r := {}
3      mid := (w^2 − 1)/2
4      half := ⌊w/2⌋
5      r_mid := c
6      for i := 0 to half do
7          r_{mid+i·w} := Rot_w(r_{mid+(i−1)·w})
8          r_{mid−i·w} := Rot_{−w}(r_{mid−(i−1)·k})
9      for i := 0 to w do
10         row := w · i
11         for j := 0 to half do
12             idx := row + half + (j + 1)
13             r_idx := Rot_1(r_{idx−1})
14             idx := row + half − (j + 1)
15             r_idx := Rot_{−1}(r_{idx−1})
16     return r
```

Algorithm 5: Procedure to generate the rotations

The number of required rotations for this phase remains w^2-1, but the number of automorphism keys is constant and equal to four, whereas in previous works [Lee+22b] based on Vector encoding convolutions, the number of rotation keys is equal to $w^2 - 1$.

2) Applying a set of filters. Given a set of w^2 rotations, each rotation must be multiplied by the corresponding weight. Intuitively, the resulting w^2 ciphertexts are summed, and the result will contain the output of the filter application.

A kernel K is defined as a four-dimensional tensor $K \in \mathbb{R}^{n_{out} \times n_{in} \times w \times w}$ containing n_{out} filters, one for each output channel. Each filter is a three-dimensional tensor $\mathbb{R}^{n_{in} \times w \times w}$ in which n_{in} is the number of input channels.

The intuition is that each rotation c_i represents the i-th element on which the filter is applied. For instance, c_0 represents the values that will be multiplied with the first element of the filter window (upper-left), c_1 with the second element of the filter window (upper), and so on.

We therefore create w^2 plaintexts containing w^2 repetitions of the corresponding kernel value. Considering the example filter presented in Fig. 72, we encode w^2 plaintexts $\boldsymbol{k}_i$ as shown in Figure 75.

This method encodes a $w \times w$ filter by using w^2 slots in w^2 plaintexts. The w^2 repetitions are necessary as they allow the

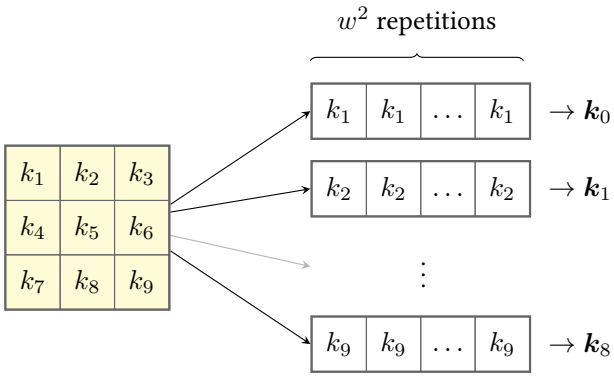


Figure 75: Encoding a filter in nine plaintexts $\boldsymbol{k}_i$

parallel evaluation of the application of a filter to an entire channel, which is composed of w^2 values. Some values will be masked, in order to avoid to align values coming from other channels (notice gray values in Figure 74), we therefore put zeros in those slots. We can thus compute a temporary ciphertext **t** as shown in Figure 76.

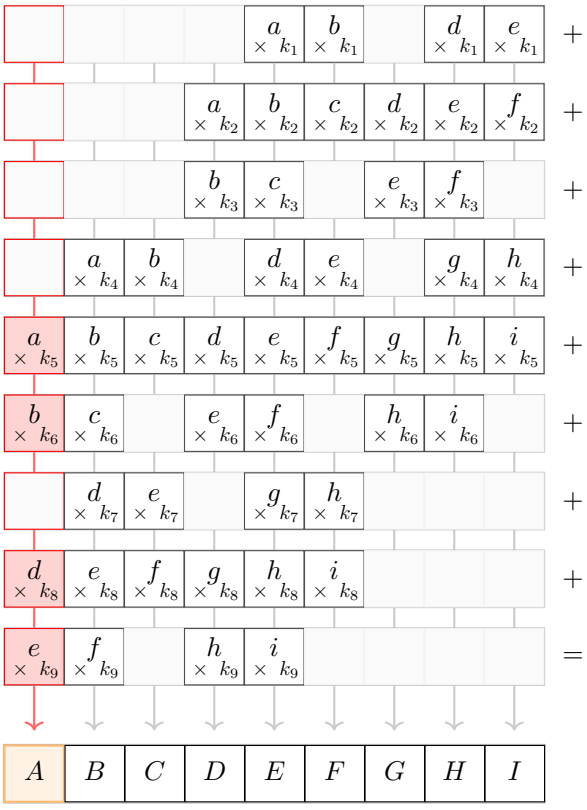


Figure 76: Evaluating a convolution by adding k^2 ciphertexts up, following the example in Figure 72

More formally, each line computes $\boldsymbol{r}_i \cdot \boldsymbol{k}_i$ and $\boldsymbol{t} := \sum_{i=0}^{8} \boldsymbol{r}_i \cdot \boldsymbol{k}_i$ is the sum of the nine lines. Nevertheless, there are a lot of empty slots in the plaintexts $\boldsymbol{k}_i$, since we are aligning a kernel in the first w^2 slots only. We extend the procedure to apply c filters over c channels by arranging the values as shown in Figure 77.

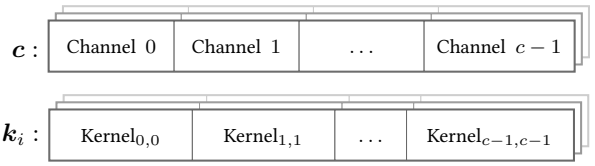
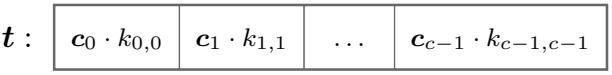
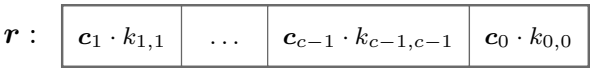


Figure 77: Elements alignment before the first iteration

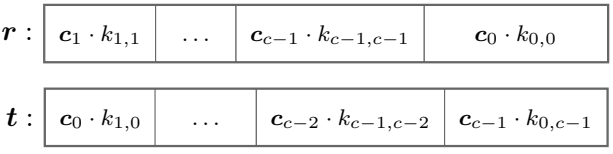
We thus obtain a single ciphertext $\boldsymbol{t}$ in which each i-th block (consisting of w^2 elements) contains the application of the i-th filter on the i-th channel, arranged as follows:



3) Applying the whole kernel. Previously, we applied a single filter to each channel. To complete the convolution process, all the n_{out} filters of the kernel need to be applied to each of the n_{in} input channels. As a first step, we rotate by w^2 the ciphertext $\boldsymbol{t}$, obtaining $\boldsymbol{r} = \text{Rot}_{w^2}(\boldsymbol{t})$:



In the second iteration, we apply the second two-dimensional convolutional window, obtaining a new $\boldsymbol{t}$, and add it to $\boldsymbol{r}$.



Notice that the index of the ciphertext $\boldsymbol{c}$ and the first index of K are always aligned. At the last iteration, the first channel will be back to the first block, since we rotated by w^2 for c

times, which is the length of the ciphertext. Our procedure sums the following ciphertexts up:

$\boldsymbol{r}$:	$\boldsymbol{c}_1 \cdot k_{1,1}$	...	$\boldsymbol{c}_{c-1} \cdot k_{c-1,c-1}$	$\boldsymbol{c}_0 \cdot k_{0,0}$
$\boldsymbol{t}$:	$\boldsymbol{c}_0 \cdot k_{1,0}$	...	$\boldsymbol{c}_{c-2} \cdot k_{c-1,c-2}$	$\boldsymbol{c}_{c-1} \cdot k_{0,c-1}$

We define the algorithm used for the generation of the encoded kernel plaintext in Algorithm 6, and the complete procedure for the Optimized Vector Encoding in Algorithm 7.

```
function EncodeKernel (K, iter)
    k := empty list
    for i := 0 to n_out do
        filters := {}
        for j := 0 to k^2 do
            idx := (i + iter) mod n_in
            rep := repeat(K_{i,idx,j}, w^2)
            filters := append(rep)
        k := append(filters)
```

Algorithm 6: Procedure that encodes the kernel

In usual Vector Encoding, each iteration requires a different key, for a total of c keys. We thus improved the keys requirements for Vector Encoding from $(k^2 - 1 + c)$ to a constant value 5 ($\approx 94\%$ of memory usage reduction in Layer 3, when $c = 64, k = 3$).

```
function OptimizedVectorEncoding (K, k)
    r := 0
    c := GenerateRotations(c, k)
    for i := 0 to n_out do
        k := EncodeKernel(K, i)
        t := 0
        for j := 0 to k^2 do
            t := t + (c_j · k_j)
        r := r + Rot_{w^2}(t);
```

Algorithm 7: Optimized Vector Encoding

5.2.3 ResNet20 Circuit

Recalling the structure of a ResNet20 (Section 5.2.1), we want to build an equivalent sequence of blocks based on FHE. The first issue is about the position of bootstrapping operations. We consider a Basic block (Figure 71), and we build the FHE equivalent as shown in Figure 78. Notice that we merged Convolutional layer and Batch normalization in a single ConvBN layer (refer to Section 5.2.2) that consumes one level. We fix the level of the input ciphertext at $\ell - 1$. ConvBN is therefore evaluated before bootstrapping. In an example shown in Figure 78 we assume six levels before bootstrapping, but this, in general, depends on the degree d of the ReLU Chebyshev Polynomial approximation.

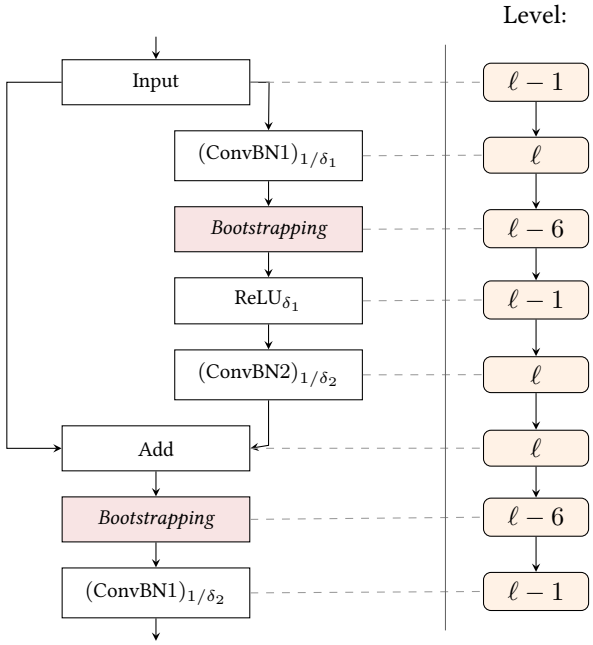


Figure 78: FHE-based structure of a ResNet block. The boxes on the right represent the ciphertext modulus level, assuming a circuit depth ℓ. Here we assume that the ReLU consumes five levels

5.2.3.1 ReLU Approximation Interval

When using Chebyshev Polynomials in order to approximate a non-linear function, an approximation interval must be defined. Using an interval different from $[-1, 1]$ is not convenient as it requires an additional level consumption, since a generic $[a, b]$ must be scaled down to $[-1, 1]$.

Our approach is to study the range $[-\delta_i, \delta_i]$ of values before all the ReLU evaluations in the plain ResNet20, and to run the

ConvBN with a scale $1/\delta_i$ (i.e. $\text{ConvBN}_{1/\delta_i}$) such that all the inputs given to ReLU are in $[-1, 1]$:

$$\text{ReLU}_{\delta_i}\left(\text{ConvBN}_{1/\delta_i}\right) = \begin{cases} \delta_i\, x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

We ended up with the experimental values for the approximating factors δ_i, $i \in \{1, 2\}$, in each block of each layer, reported in Table 8.

Layer	Block 1		Block 2		Block 3	
	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2
1	1.00	0.52	0.55	0.36	0.63	0.42
2	0.57	0.40	0.76	0.37	0.63	0.25
3	0.63	0.40	0.57	0.33	0.69	0.10

Table 8: Experimentally obtained values for δ_i's, the approximating factors for intervals in ReLU

5.2.3.2 Downsampling Layers

Given that the total number of values decreases during the evaluation of the network (refer to Table 7), it is possible to reduce the number of slots in ciphertexts accordingly. In particular, downsamplings are performed by evaluating a Convolution Layer with stride $\{2, 2\}$ in Downsampling blocks (Figure 70). Our Optimized Vector Encoding method is no longer sufficient, since it will blank slots in the output ciphertexts. A reshaping operation is therefore needed in order to fill these empty spaces. This process is performed by consuming six levels, and it intuitively works as shown in Figure 79.

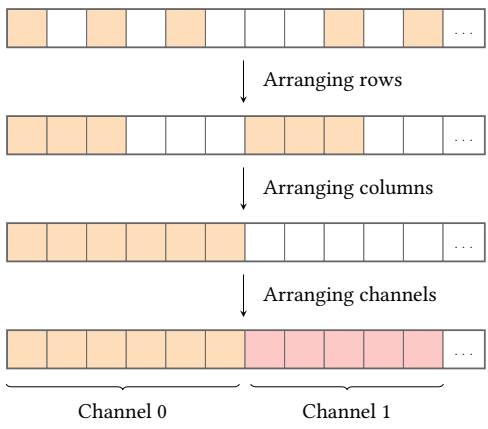


Figure 79: Reshaping values after a Convolutional Layer with stride $\{2, 2\}$

Re-arranging values requires new rotation keys. Assuming that the ciphertext has s slots, our strategy is to perform this procedure when the bootstrapping keys for s slots are no longer needed, so that they can be cleared from the memory. We can then load the keys needed for the downsampling, perform the procedure and set the number of slots for the ciphertexts to $s/2$. After that, we load the new bootstrapping keys for $s/2$ slots. The following layer will be evaluated with smaller ciphertexts, although with twice the number of channels.

5.3 Experiments and evaluations

We now present the experiments run using four sets of parameters. Each experiment will be evaluated according to three main factors: computations precision, computation time and memory requirements. The precision of the FHE circuit is evaluated by running a parallel circuit working on plain values; we evaluate it, considering that CKKS works using fixed-point arithmetic, as:

$$p(\boldsymbol{v}, \boldsymbol{v}') = 1 - \left(\sum_{i=1}^{\text{len}(\boldsymbol{v})} \left(\frac{|v_i - v'_i|}{|\max(\boldsymbol{v})|} \right) \cdot \|\boldsymbol{v}\|^{-1} \right) \tag{5.3}$$

where $\boldsymbol{v}$ and $\boldsymbol{v}'$ are two vectors, output of the plain and the FHE circuit, respectively. The considered CIFAR-10 test set is composed of 1000 test images of size 32×32, each representing one out of 10 classes.

5.3.1 Circuit parameters

All the experiments have been run on 1.000 images from the CIFAR-10 test set, with the constraint of using less than 16GB of RAM and to satisfy the level of $\lambda \geq 128$ security bits according to the *Homomorphic encryption security standard* [Alb+18].

Some common RLWE parameters for all the experiments are the following: Hamming weight (the number of non-zero elements in the secret key) set to $h = 64$, standard deviation of error distribution $\sigma = 3.19$. On the other hand, each experiment has different values for the following parameters:

- Precision (Δ, q_i, d) : this set of parameters has a big impact on the precision of the result. However, larger values mean slower and heavier computations.

Table 9: Parameters for the CKKS scheme, grouped by experiment.

Exp	Ring (n)	Scaling factor (Δ)	Size of q_i	HKS digits (d_{num})	$\log(QP)$	Degree ReLU (d)	Circuit depth (ℓ)	`CtoS` and `StoC` levels
1	2^{16}	2^{52}	2^{48}	2	1756	59	23	$\{3, 3\}$
2	2^{16}	2^{50}	2^{46}	3	1772	200	27	$\{4, 4\}$
3	2^{16}	2^{50}	2^{46}	3	1772	119	27	$\{5, 4\}$
4	2^{16}	2^{48}	2^{44}	2	1748	59	26	$\{4, 4\}$

► Memory $(n, q_i, d_{num}, \ell, \texttt{CtoS}, \texttt{StoC})$: these parameters impact the memory requirements. In particular, the size of each rotation key is roughly:

$$2\ell d_{\text{num}} N(1 + \frac{1}{d_{\text{num}}}).$$

Also, the levels dedicated to the `CoeffToSlot` and the `SlotToCoeff` are important, because larger values mean less complex bootstrapping.

► Time $(n, q_i, d, \texttt{CtoS}, \texttt{CtoS})$: the first parameter, n, has the heaviest impact in computation time, as it defines the magnitude of all calculations. Smaller q_i and d result in smaller ciphertexts (hence, faster computations).

The output of the circuit is a vector containing ten values representing the activation of each output neuron, each associated with an output class. A sample successful output is presented in Figure 80. When the difference between the first

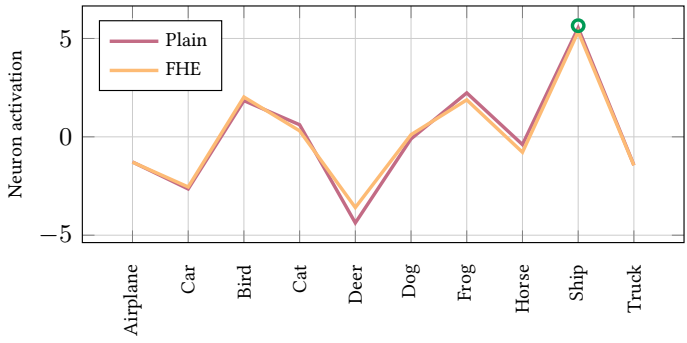


Figure 80: An example of successful FHE output

and the second most active neurons is large, even approximate computations are successful. On the other hand, approximate classifications fail when the error changes the index of the maximum value (Figure 81). This is likely to happen when the difference between the first and second most active neurons is small. We studied how the difference between the first and

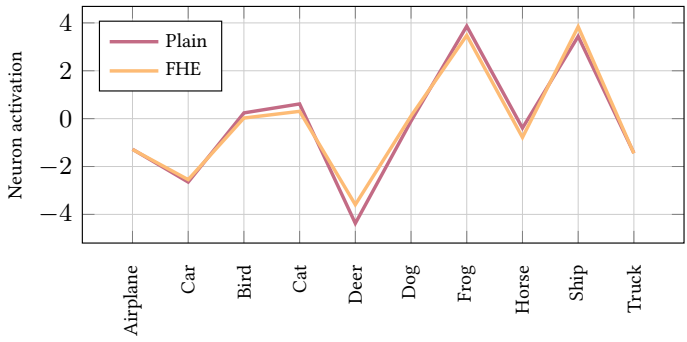


Figure 81: Example of wrong FHE classification with respect to the plain one (Frog for plain, Ship for FHE)

second most active neurons is distributed, considering the complete set of images from the CIFAR-10 test set, using the plain ResNet20 model[1]. Given y_i, the sorted vector containing the classification of the i-th image, we define a vector $\boldsymbol{d}$:

$$\boldsymbol{d} := \{d_i : d_i = (y_i)_0 - (y_i)_1\}.$$

The mean value of differences is ≈ 3.12, with a standard deviation $\sigma(\boldsymbol{d}) \approx 2.64$. In general, more than 98% of the images have a difference d_i that is larger than 10^{-1}. This means that, when approximation errors are less than 10^{-1}, at least 98% of the results is equal to the plain model.

All the results presented in the experiments are obtained by evaluating the corresponding FHE circuit on a set of one thousand encrypted images from the CIFAR-10 test set, using a M1 Pro CPU and 16GB of RAM. The set of parameters for each experiment is given in Table 9.

It is possible to reproduce all the experiments using our open-source code[2] in C++. Algorithm 5 and 6 are available as Python notebooks, along with a notebook that shows how values from Table 8 have been found. As far as we are aware, ours and [KG23b] (written using the Lattigo [Lat24] library, in Go language) are the only recent and available open-source FHE-based CNN implementations.

1: github.com/chenyaofo/pytorch-cifar-models

2: github.com/lorenzorovida/LowMemoryFHEResNet20

[KG23b]: D. Kim and Guyot (2023), *'Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption'* . In: IEEE Transactions on Information Forensics and Security

[Lat24]: (2024), Lattigo v6, *EPFL-LDS, Tune Insight SA* (Online: `https://github.com/tuneinsight/lattigo`)

5.3.2 Discussion

We present all the results obtained by the four experiments in Table 10 and in Figure 82.

Experiments show interesting results. To begin with, all the considered sets of parameters result in at least 0.93 of relative accuracy. Since the original model has a level of accuracy of

Table 10: Comparison of results obtained in the experiments. Relative accuracy is computed with respect to the plain results on a subset of 1000 images from the test set

Exp	Runtime	Bootstrapping precision	Memory requirements	Output precision	Relative accuracy	Final accuracy
1	$291s \pm 4s$	≈ 9.4 bits	≈ 13.6GB	0.93 ± 0.02	$\approx 0.95\%$	87.97%
2	$336s \pm 6s$	≈ 7.7 bits	≈ 15.2GB	0.98 ± 0.01	$\approx 0.99\%$	**91.67**%
3	$285s \pm 6s$	≈ 7.7 bits	≈ 12.5GB	0.97 ± 0.01	$\approx 0.98\%$	90.75%
4	$260s \pm 3s$	≈ 7.6 bits	$\approx$ **11.6GB**	0.92 ± 0.02	$\approx 0.93\%$	86.12%

0.92, this results in at least 0.85 of final accuracy, using any set of parameters. The set relative to Experiment 4, in particular, is the one having the lightest memory requirements (i.e. only 11.6GB) and the lowest runtime (less than 5 minutes).

By increasing the complexity of the parameters, we find the set of Experiment 1 to be slightly more complex than the previous one. In particular, we increased the values of Δ and q_i for a higher accuracy in computations. In order to maintain the required level of security, we reduced by one the levels dedicated to `CtoS` and `StoC`. As a result, we obtained an increase on the relative accuracy ($+0.02$) at the cost of 2 more GB of memory and a longer runtime ($+7s$).

Surprisingly, with the set of parameters defined in Experiment 3, we obtained a larger relative accuracy (0.98) even though the required memory is increased by only 1GB with respect to Experiment 4. This confirms that the main bottleneck in computations precision is the degree d of the ReLU polynomial approximation. We then furthermore increased the degree to $d = 200$ in Experiment 2, obtaining the highest level of relative accuracy (0.99). This costs in terms of memory, since it requires around 15.2GB. Computational times, too, are higher.

To recap, we propose Experiment 3 as the one having the best ratio between precision of computations, memory requirements and execution time.

5.4 Conclusion

We presented a FHE-based circuit that classifies encrypted images with a high level of accuracy (0.9167) on the CIFAR-10 test set, evaluable in less than five minutes on a MacBook laptop equipped with M1 Pro CPU and 16GB of memory. This reduces the gap between current FHE circuits and real-world

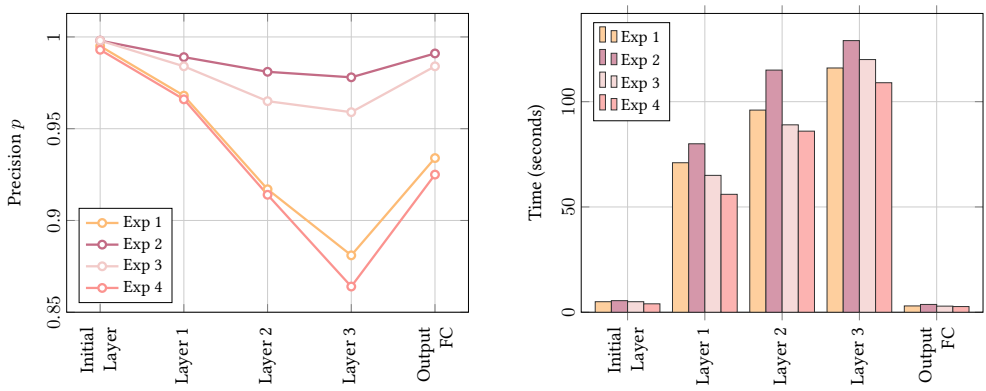


Figure 82: Visual representation of precision and runtime of the experiments for the encrypted circuit

applications. In particular, we demonstrated that a server is able to offer a private image recognition service using a small amount of memory, which has been demonstrated to be a big bottleneck when deploying HE applications [Cas+21]. The order of magnitude of the achieved accuracy is similar to other recent works [KG23b; Lee+22b], but our approach has a reduction of memory requirements by about 85% from Kim *et al*. [KG23b] and by about 98% from Lee *et al*. [Lee+22b].

Execution times are lower, although it is not easy to compare circuits executed on different hardware. Also for this reason, our work is available open-source. As far as we are aware, ours and Kim and Guyot [KG23b] are the only CNNs based on FHE available open-source, as other recent works do not share any source code.

Furthermore, the circuit can be made even more suitable for real applications by using dedicated hardware [Lat+23; Sam+22], which allows for extreme speed-up for FHE circuits. Theoretically, a 4.600x speed-up [Sam+22] would result in approximately five encrypted classifications per second using our circuit, enabling privacy-preserving inferences to be used in real-world tasks.

[Cas+21]: Castro, Agrawal, Yazicigil, Chandrakasan, et al. (2021), Does Fully Homomorphic Encryption Need Compute Acceleration?

[KG23b]: D. Kim and Guyot (2023), *'Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption'* . In: IEEE Transactions on Information Forensics and Security

[Lee+22b]: E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, et al., *'Low-Complexity Deep Convolutional Neural Networks on Fully Homomorphic Encryption Using Multiplexed Parallel Convolutions'* . In: PMLR '22

[Sam+22]: Samardzic, Feldmann, Krastev, Manohar, et al., *'CraterLake: A Hardware Accelerator for Efficient Unbounded Computation on Encrypted Data'* . In: ISCA '22

Transformers architecture over encrypted text

6

This chapter is an extended/revised version of 'Transformer-based Language Models and Homomorphic Encryption: An Intersection with BERT-tiny' *co-authored with A. Leporati, published in the proceedings of IWSPA '24*

Abstract: In recent years, emerging and improved Natural Language Processing (NLP) models, such as Bidirectional Encoder Representations from Transformers (BERT), have gained significant attention due to their performance on several natural language tasks. However, inappropriate focus is usually given to the critical problems of security and data privacy, since these models require access to plain data. To address these issues, we suggest a solution based on Fully homomorphic encryption (FHE), which allows for computations to be performed on encrypted data. In particular, we propose an FHE-based circuit that, by implementing the smallest existent BERT model, namely $\text{BERT}_{\text{tiny}}$, enables the extraction of encrypted sentences representations and encrypted text classifications. Considering the nature and the depth of this circuit, we used the Cheon-Kim-Kim-Song (CKKS) scheme, along with the bootstrapping operation. We also propose to use precomputations for the Layer normalization, in order to lighten computations. The experiments, which can be replicated using our open-source code, are conducted on the Stanford Sentiment Treebank (SST-2) dataset. They show that errors introduced by precomputed Layer Normalizaion, approximate FHE operations and polynomial approximations do not produce a significant performance loss.

6.1 Introduction

Bidirectional Encoder Representations from Transformers (BERT) [Dev+19] is a revolutionary Natural language processing (NLP) model that employs the Transformer [Vas+17b] architecture. One of the key features of Transformers is the Attention mechanism, which allows to capture contextual information about words and to focus on different relationships among them. Models based on Attention have demonstrated very high performance in many NLP

[Dev+19]: Devlin, Chang, K. Lee, and Toutanova, *'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'* . In: NAACL '19

[Vas+17b]: Vaswani, Shazeer, Parmar, Uszkoreit, et al., *'Attention is All You Need'* . In: NIPS '17

tasks, since they overcome the previous limitations of traditional sequential models. Despite this, insufficient attention is currently given to ensuring data security and privacy.

In this context, (FHE) emerges as a possible solution, as it allows for computations to be performed on encrypted data. Many FHE schemes have been proposed in the literature, each having advantages and drawbacks. In particular, the CKKS scheme [CKKS17] well-suits the context of neural networks [CSZ21] since it works on complex numbers and has the very beneficial property of batching multiple values in a single ciphertext. It therefore takes advantage of fast SIMD operations, meaning that additions and multiplications are performed slot-wise between encrypted vectors, leading to fast parallel computations. In particular, in this chapter we propose a circuit based on the OpenFHE library [Al +22], which offers an optimized implementation of the RNS-CKKS scheme [KPP22].

Since FHE adds significant overhead in computations, we propose an implementation of the smallest existing BERT model, called $\text{BERT}_{\text{tiny}}$ [BDR21]. It is a scaled-down version of the BERT model, which consists of $L = 2$ encoders of $H = 128$ hidden units, designed to be more compact and computationally efficient compared to the original BERT. Such efficiency is obtained through the so-called distillation process [Tur+20], that trains the smaller model ($\text{BERT}_{\text{tiny}}$) using the outputs or representations of the larger model (BERT) as "teacher" signals.

[CSZ21]: Clet, Stan, and Zuber, *'BFV, CKKS, TFHE: Which One is the Best for a Secure Neural Network Evaluation in the Cloud?'* . In: ACNS '21

[Al +22]: Al Badawi, Bates, Bergamaschi, Cousins, et al., *'OpenFHE: Open-Source Fully Homomorphic Encryption Library'* . In: WAHC '22

[BDR21]: Bhargava, Drozd, and Rogers (2021), *'Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics'* . In: Proceedings of the Second Workshop on Insights from Negative Results in NLP

[Tur+20]: Turc, Chang, K. Lee, and Toutanova, *'Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation'* . In: ICLR '20

6.1.1 Related works

The intersection between NLP and FHE schemes has been somewhat explored in the last years. In this section we review the most relevant works that explore this intersection, highlighting the differences with respect to our approach. The main issue, currently unsolved, is the overhead introduced by HE computations, which prevents Large language models (LLMs) to be implemented.

In 2020, [Bad+20] proposed PrivFT, perhaps the first work that showed practical text classification using the *fasttext* [Jou+17] architecture and the first RNS variant for CKKS accelerated by GPU. The implementation showed very good performance. Nevertheless, the *fasttext* architecture is much simpler than Transformers, and the evaluation of softmax function is performed by the client.

[Bad+20]: Badawi, Hoang, Mun, Laine, et al. (2020), *'PrivFT: Private and Fast Text Classification With Homomorphic Encryption'* . In: IEEE Access

[Jou+17]: Joulin, Grave, Bojanowski, and Mikolov, *'Bag of Tricks for Efficient Text Classification'* . In: EACL '17

A couple of years later, [Che+22] introduced THE-X, a HE implementation of the Transformer architecture. The main problems of this approach are the introduction of many approximations, simplified computations, and the workflow in which the client has to get involved in some computations (especially in the evaluation of some activation functions). One of our main contribution is, on the other hand, to outsource all the computational tasks to the server.

[Che+22]: T. Chen, Bao, S. Huang, Dong, et al., *'THE-X: Privacy-Preserving Transformer Inference with Homomorphic Encryption'* . In: ACL '22

Recently, [ZLJ25] introduced Primer, a privacy preserving architecture which allows fast Transformer inference. In that work, HE is used for polynomial operations and MPC for non-polynomial operations, so the whole architecture is hybrid and involves multiple actors. Our architecture, on the other hand, involves only a service provider and a client.

[ZLJ25]: Zheng, Lou, and Jiang, *'Primer: Fast Private Transformer Inference on Encrypted Data'* . In: DAC '23

One key point in our work is the evaluation of the softmax function, which requires the evaluation of e^x and $1/x$. In [Hon+22], the exponential function is evaluated using the limit definition by Euler, and the inverse function is approximated using Goldschmidt algorithm. In [Lee22] e^x is approximated using the least-squares method and, as before, Goldschmidt algorithm is applied. Our proposal takes instead advantage of the Maclaurin series for e^x and of Chebyshev polynomial for $1/x$, which provides a polynomial approximation close to the best [Tre19].

[Hon+22]: S. Hong, J. H. Park, Cho, Choe, et al. (2022), *'Secure tumor classification by shallow neural network using homomorphic encryption'* . In: BMC Genomics

[Lee22]: J.-W. e. a. Lee (2022), *'Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network'* . In: IEEE Access

[Tre19]: Trefethen (2019), Approximation Theory and Approximation Practice, Extended Edition

As far as we are aware, the only intersection between HE and BERT is explored in [Lee+22d], where a HE-based Logistic Regression model is trained using BERT embeddings. Nevertheless, BERT is only slightly involved in such work, since the main goal is the implementation of the Logistic Regression model. The inference from BERT, on the other hand, is performed by the client, and not by the service provider, as in our proposal.

[Lee+22d]: G. Lee, M. Kim, J. H. Park, Hwang, et al., *'Privacy-Preserving Text Classification on BERT Embeddings with Homomorphic Encryption'* . In: NAACL '22

6.1.2 Our contribution

In this work, the following points are explored and presented:

► An implementation of the BERT encoder using FHE primitives, executed exclusively by the server. The initial embeddings, obtained as values from a look-up table, are computed by the client. To the best of our knowledge, this is the first BERT circuit based on FHE.

► A *precomputation* of Layer normalization, in which precomputed values for mean and variance are used in order to reduce the complexity of the circuit. We demonstrate that this approach does not result in a significant reduction in the model performance.
► A performance evaluation of the encrypted circuit, fine-tuned on the Stanford Sentiment Treebank (SST-2) [PL05] dataset, which is used for a sentiment analysis task.

6.1.3 The proposed setting

We propose a setting composed of two parties: a server that offers a machine learning service based on BERT and a client (or user) which asks for a classification. The communication between the server and the client (or the user) is described by Figure 83.

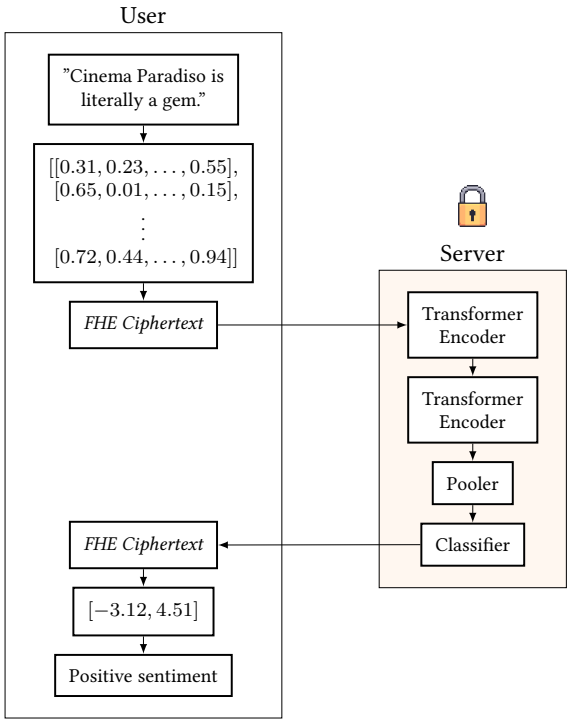


Figure 83: High-level architecture of our proposal

The client computes a set of embeddings of a tokenized sentence using look-up tables. This set is encrypted and the ciphertexts are sent to the server. The latter evaluates a HE circuit which implements two $\text{BERT}_{\text{tiny}}$ encoders, a pooling layer and a binary classification layer. The result is thus a

ciphertext containing two values, one for each output neuron. The client is able to get the result of the classification by decrypting the ciphertext and verifying whether the value contained in the first slot is greater than the value contained in the second slot.

Approximate homomorphic encryption. Our circuit is based on the CKKS [CKKS17] scheme, which is an approximate HE scheme that allows computations on encrypted vectors of complex numbers. We briefly recall the main functionalities of the scheme, but we refer to Section 3.3 for further details.

Let $\mathcal{R} := \mathbb{Z}[X]/(\mathbb{Z}^n + 1)$ be a cyclotomic ring for a power of two n, and $\mathcal{R}_Q$ be the quotient ring of $\mathcal{R}$. Given a plain vector $\mathbf{v} \in \mathbb{C}^{N/2}$, the encryption informally follows the following process:

$$\mathbf{v} \in \mathbb{C}^{N/2} \xrightarrow{\text{encode}} \mathcal{R} \xrightarrow{\text{encrypt}} \mathcal{R}_Q^2$$

Since values are discretized in $\mathbb{Z}$, a parameter Δ is introduced in order to control the precision of the encoding procedure. Vectors are encoded in polynomials in such a way that polynomial multiplications result in slot-wise multiplication in the clear space $\mathbb{C}^{N/2}$.

We refer to the *level* of a ciphertext as the number of multiplications that has been performed on it. A multiplication between two ciphertexts result in a ciphertext with a doubled scale equal to Δ^2; therefore, a so-called *rescaling* operation is performed. Nevertheless, this operation reduces the modulo Q of the polynomial by multipliying by a factor Δ^{-1}.

The modulus $Q = q_1 \cdot q_2 \cdot \ldots q_n$ is built as a *moduli chain* using multiple q_i, so that when a rescaling is performed, the modulo Q "loses" a modulus q_i. When Q reaches the minimum level $Q = q_1$, it can not be rescaled anymore. A *bootstrapping* operation is required in order to bring back the modulus to the original Q (equivalently, to reduce the level of a ciphertext). We use the bootstrapping technique described in [Bou+21].

This chapter uses the following primitives, using OpenFHE API notation:

- EvalAdd($\boldsymbol{c}, \boldsymbol{c}'$): performs a slot-wise addition between two ciphertexts/plaintexts $\boldsymbol{c}$ and $\boldsymbol{c}'$.
- EvalMult($\boldsymbol{c}, \boldsymbol{c}'$): performs a slot-wise multiplication between two ciphertexts/plaintexts $\boldsymbol{c}$ and $\boldsymbol{c}'$.
- EvalRotate($\boldsymbol{c}, i$): the positions of the encrypted values in $\boldsymbol{c}$ are rotated to the left by i positions.

[Bou+21]: Boussat, Mouchet, J. Troncoso-Pastoriza, and Habaux, *'Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys'* . In: EUROCRYPT '21

► EvalBootstrap($\boldsymbol{c}$): performs the bootstrapping operation on the ciphertext $\boldsymbol{c}$.

Since the bootstrapping operation theoretically enables the evaluation of circuits of any depth, the term Fully homomorphic encryption (FHE) is used.

6.2 Methodology

This section contains the design of the proposed FHE circuit, and the definition of the basic algorithms on which the circuit is based.

6.2.1 Some basic HE algorithms

Efficient computations in a HE circuit, particularly matrix multiplications, require adequate methods for encoding and packing data. In literature, it is possible to find many approaches to vector-matrix multiplication. For instance, Halevi and Shoup [HS14] presented a so-called diagonal form for multiplications, while Juvekar, Vaikuntanathan and Chandrakasan [JVC18b] an even better hybrid approach.

When building an encrypted version of a feed-forward or convolutional neural network, it is usually possible to apply any pre-processing (i.e., moving and repeating elements) to weight matrices, since they are almost always stored as plaintexts. Nevertheless, in the Self-attention layer of Transformers, many matrix multiplications are performed between matrices that can not be pre-processed. For instance, the first multiplication is performed between $\mathbf{Q}$ and $\mathbf{K}$ which are, in turn, obtained as matrix multiplications. In this case pre-processing is not possible, and reshaping ciphertexts is an expensive operation. We therefore propose two algorithms to perform matrix-vector multiplication, depending on how the arguments are packed. The main idea is that, in order to avoid reshaping, procedures are adapted to data, and not vice versa. Since $\text{BERT}_{\text{tiny}}$ works mostly on $H = 128$ hidden units, the most common operation is between matrices of dimension 128×128 and vectors of length 128. As a consequence, this work is based on plaintexts[1] (and ciphertexts) containing $128 \cdot 128 = 16384 = 2^{14}$ values (so that we can encode the whole weight matrix in a single plaintext). Given a vector of length k, and ciphertexts of s slots, where $k < s$ and both are powers of two, we define:

1: For the sake of clarity, the plain matrix is called *Cleartext*, the matrix encoded in a polynomial (thus, usable in computations) is called *Plaintext*, while the encryption function gives as output a *Ciphertext*.

- Repeated packing: the plain vector is repeated k/s times along the ciphertext.
- Expanded packing: each of the k elements of the plain vector is repeated k/s times.

This is a *required redundancy*, needed to take full advantage of SIMD computations. Notice that in our implementation k is equal to the hidden size H of the model (in $\text{BERT}_{\text{tiny}}$ the hidden size is $H = 128$), while the number of slots in each ciphertext is $s = 2^{14}$. Matrices, on the other hand, are encoded in row-major packing or in column-major packing.

Vector-matrix multiplications. We introduce two distinct procedures, namely MatVecEC and MatVecRC. Both work on vectors of length n and square matrices of size $n \times n$, and they share the same semantic. Plus, they both rely on the RotSum procedure, that given b and t as input, sums the elements in positions $\{i \cdot b : 0 \leq i < t\}$ by rotating the input ciphertext. The first one, VecMatER, is performed between an Expanded vector and a Row-major matrix (as the name suggests), and is presented in Algorithm 8.

1 **function** MatVecER $(\boldsymbol{c}, \boldsymbol{c}')$
2 $\quad \boldsymbol{c}_{\text{res}} := \text{EvalMult}(\boldsymbol{c}, \boldsymbol{c}')$;
3 $\quad \boldsymbol{c}_{\text{res}} := \text{RotSum}(\boldsymbol{c}_{\text{res}}, t = 128, b = 128)$;
4 $\quad$ **return** $\boldsymbol{c}_{\text{res}}$

Algorithm 8: Vector-matrix in expanded and row-major shapes multiplication

Figure 84 provides a visual representation of the procedure.

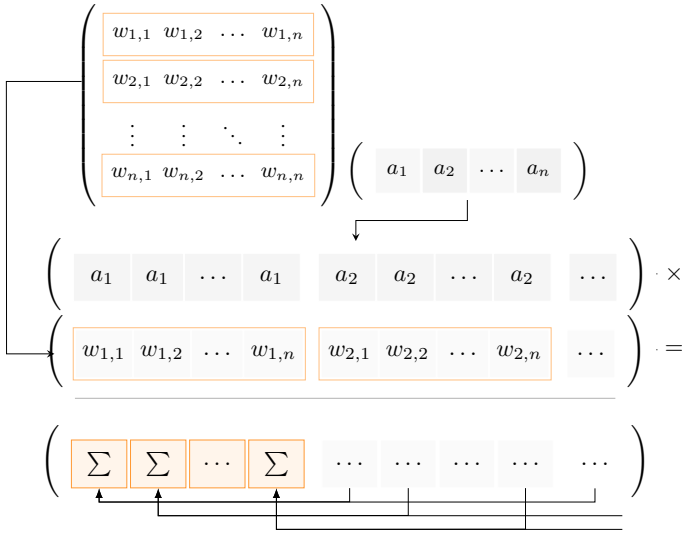


Figure 84: Visual representation of the VecMatER procedure

The procedure performs only one multiplication, meaning that only one level is consumed. The output is a ciphertext in Repeated shape, ready to be used by the following procedure, without any reshaping. The MatVecRC procedure takes as input a Repeated vector and a Column-major matrix. It is described in Algorithm 9 and it is represented in Figure 85.

function MatVecRC $(\boldsymbol{c}, \boldsymbol{c}')$
  $\boldsymbol{c}_{res} := \text{EvalMult}(\boldsymbol{c}, \boldsymbol{c}')$;
  $\boldsymbol{c}_{res} := \text{RotSum}(\boldsymbol{c}_{\text{res}}, t = 128, b = 1)$;
  $\boldsymbol{c}_{res} := \text{Repeat}(\boldsymbol{c}_{\text{res}}, t = 128, b = 1)$;
  return $\boldsymbol{c}_{res}$

Algorithm 9: Vector-Matrix in Repeated and Column-major shapes multiplication

The output of this algorithm is a ciphertext that contains the resulting values in positions $i : i \bmod 128 = 0$. By repeating these values 128 times, we obtain an Expanded ciphertext, that can be ideally used again by the MatVecER procedure. Since the repetition of values requires a mask procedure, this procedure consumes a total of two levels.

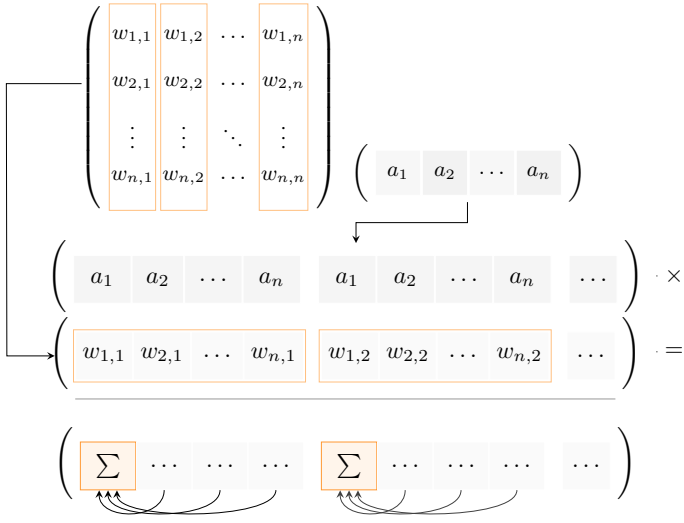


Figure 85: Visual representation of the VecMatCR procedure

Vector wrapping algorithms. We also present two procedures that, combined with vector-matrix multiplications, enable actual matrix-matrix multiplications. The idea is to perform n times a vector-matrix multiplications and to wrap the results up in a single ciphertext. The first procedure is called WrapUpExpanded, and it is used to wrap at most n Expanded

shape ciphertexts, see Algorithm 10 and Figure 86 for a visual representation.

```
1 function WrapUpExpanded (c, n)
2     for i := 0 to len(c) − 1 do
3         m_i := MaskMod(c, i, n);
4     return EvalAddAll(m_i);
```

Algorithm 10: Wrap up Expanded

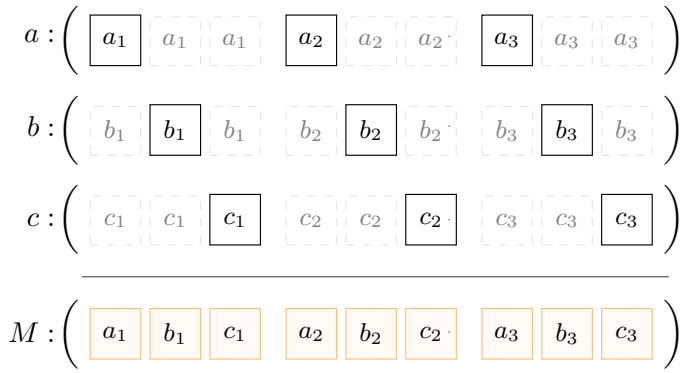


Figure 86: Visual representation of the WrapUpExpanded procedure; M is the resulting Column-major matrix

The output of this procedure is a ciphertext containing a Column-major matrix. The MaskMod procedure takes as input an index i and selects the values in positions j such that $j \equiv i \bmod n$ by applying a binary mask encoded as a plaintext. All the masked ciphertexts are then summed using the EvalAdd procedure.

On the other hand, WrapUpRepeated is used to wrap n Repeated shape ciphertexts, where n is the number of repetitions. The output is a ciphertext containing a Row-major matrix consisting of the n row vectors; see Algorithm 11 and Figure 87 for a visual representation.

```
1 function WrapUpExpanded (c, n)
2     for i := 0 to len(c) − 1 do
3         m_i := MaskBlock(c, n · i, n · (i + 1));
4     return EvalAddAll(m_i);
```

Algorithm 11: Wrap up Repeated

The MaskBlock procedure is used to extract the i-th repetition of the i-th ciphertext, as shown in Figure 87. In the actual implementation the value of n is always equal to 128.

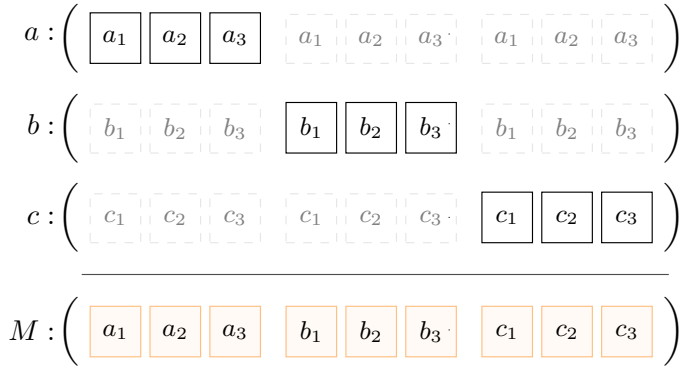


Figure 87: Visual representation of the WrapUpRepeated procedure; M is the resulting Row-major matrix

6.2.2 Circuit design

A classification model based on BERT is composed of four parts: Embeddings, Encoders, Pooler and Classifier. The first one (Embeddings) consists of a couple of lookup tables and a normalization, and is executed by the client, which stores the tables for $31036 \cdot 128$ float real numbers (a total of $\approx$ 16MB). The other three parts are homomorphically evaluated server-side, and this section introduces and analyzes them as a circuit based on FHE primitives.

Each Encoder is composed by four sequential parts: *BertSelfAttention*, *BertSelfOutput*, *BertIntermediate* and *BertOutput*.

6.2.2.1 BertSelfAttention

The first layer computes the following function:

$$\text{Self-Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6.1}$$

where Q, K and V are the query, key and value representations of the input, respectively, obtained as a result of an affine transformation $Ax + b$. The dimension d_k of the keys is, in the case of $\text{BERT}_{\text{tiny}}$, equal to 64. Lastly, $\text{softmax}(x)$ is defined as:

$$\text{softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{6.2}$$

In particular, e^x is evaluated in two phases. First, an interval $[-r, r]$ that contains all possible input values is experimentally determined. To prevent instability when working with large numbers, we approximate e^x as $e^{x/r}$ in the interval $[-1, 1]$, then the result is raised to the power of r, obtaining $(e^{x/r})^r =$

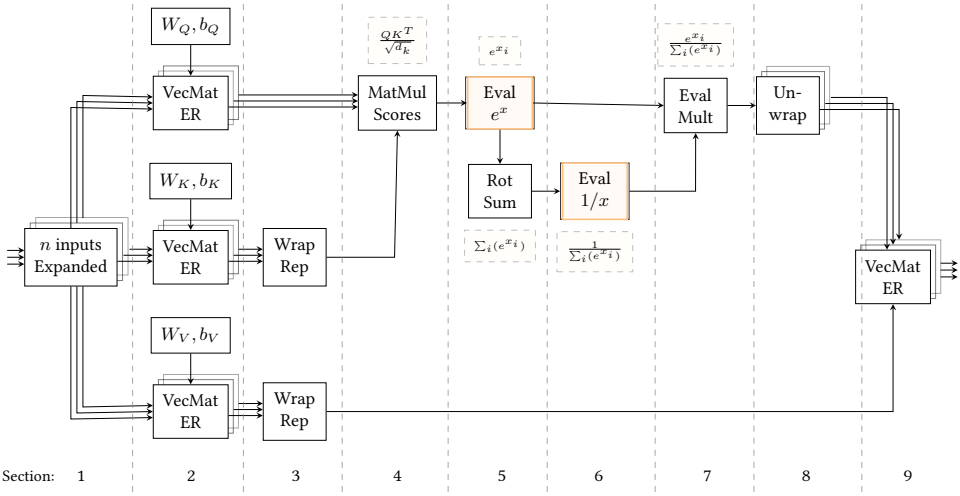


Figure 88: Circuit diagram illustrating the HE implementation of the BertSelfAttention layer

e^x. We will use n to indicate the number of input tokens. Figure 88 gives a visual representation of the HE circuit of *BertSelfAttention*, whose sections are explained below.

- *Section 1* – this is the input of the algorithm: n ciphertexts representing n token embeddings in Expanded packing shape.
- *Section 2* – the matrices Q, K and V are derived by evaluating an affine transformation $Ax+b$ using the respective weights and biases. In particular, each MatVecER operation returns a set of n ciphertexts, each representing a row of the resulting matrix.
- *Section 3* – the ciphertexts containing the rows of the key matrix K are wrapped up in a single ciphertext using the WrapUpRepeated procedure. We proceed in the same way with the rows of V. This produces two matrices encoded in Column-major order.
- *Section 4* – at this stage the so–called *scores* (in particular, the dot product attention scores) are computed. The MatMulScores procedure takes as input the matrix K, stored in a single ciphertext in Row-major packing, and the vectors q_i (with $0 \leq i < n$) containing the rows[2] of matrix Q. It computes MatVecRC(K, q_i) and wraps the results up. The masking phase is done using $(1/8) \cdot (1/r)$ as the mask value: the first term reproduces the $\sqrt{d_k}$ division, the second one prepares the values for the upcoming $e^{x/r}$ approximation.
- *Section 5* – the two heads, contained in a single ciphertext, are given as input to the first seven terms of the

2: (6.1) shows that the matrix K is transposed, but transposing is not required since the matrix is in Row-major order as a consequence of the first MatVecER procedure.

Maclaurin series of e^x. In particular, as stated above, the approximated function is $e^{x/r}$ in the interval $[-1, 1]$. The result is then raised to the power of r, consuming $\log(r)$ levels. Next, the two heads are wrapped in a single ciphertext that is then cloned: the first copy is kept as-is, while the second one is given as input to the $RotSum$ procedure. At this point, the sum of all the required terms e^{x_i} for the k-th softmax denominator (6.2) is placed in positions i such that $i \equiv k \bmod 128$, as shown in Figure 89.

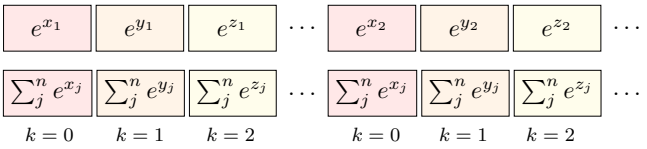


Figure 89: Example of ciphertexts content after RotSum of e^x

We use $0 \leq k < n$ to index the different denominators, since the ciphertext wraps all the n inputs.

- *Section 6* – since each softmax requires the inverse of the sum $\sum_{j=1}^n e^{x_j}$, these values are given as input to a polynomial approximation of $1/x$.
- *Section 7* – to finalize the division, the numerators e^{x_i} (the first clone) are multiplied by $1/\sum_{j=1}^n e^{x_j}$.
- *Section 8* – n vectors are unwrapped in separate ciphertexts. The UnwrapExpanded procedure does the inverse of WrapUpExpanded.
- *Section 9* – using the MatVecER procedure, the n Expanded vectors obtained by unwrapping softmax results are multiplied by V. Notice that the results will be in Repeated shape.

6.2.2.2 BertSelfOutput

This layer computes a dense layer followed by a Layer Normalization [BKH16], which is defined as:

$$\text{LayerNorm}(X) := \frac{X_i - \text{E}[X]}{\sqrt{\text{Var}[X] + \varepsilon}} \cdot \gamma + \beta \qquad (6.3)$$

Each X_i, with $0 \leq i < 128$, represents a feature of the input token X. $\text{E}[X]$ is the mean of these values, while $\text{Var}[X]$ the variance. The weight and the bias of the layer are referred to γ and β, respectively.

One of the main bottlenecks when building HE-based circuits is the evaluation of non linear functions; notice that the Layer

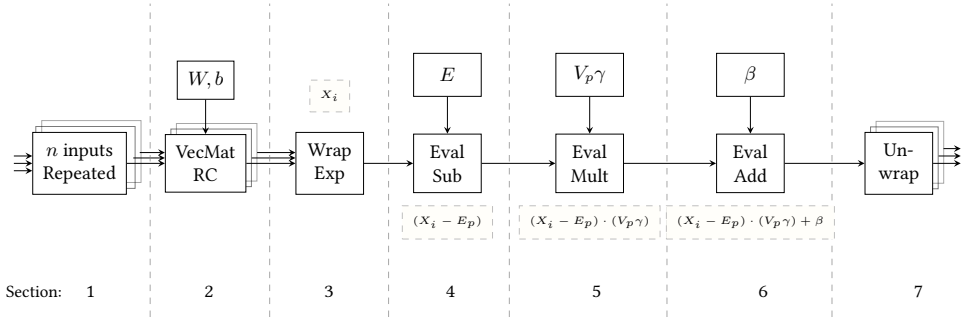


Figure 90: Circuit diagram illustrating the HE implementation of the BertSelfOutput layer

normalization implies the evaluation of an inverse square root. Taking inspiration from Residual networks [He+16a], where the Batch normalization is evaluated using precomputed values, we implemented a *Precomputed Layer normalization*, where the values of $\mathrm{E}[X]$ and $1/\sqrt{\mathrm{Var}[X] + \varepsilon}$ are experimentally observed and precomputed. We therefore computed vectors of mean values (represented as blue and red lines in Figure 91) and simplified (6.3) as follows:

$$\mathrm{LayerNormPrec}(X) := \big(X_i - E_p\big) \cdot \big(V_p\gamma\big) + \beta \qquad (6.4)$$

where E_p is the precomputed mean vector for $\mathrm{E}[X]$, and V_p is the precomputed mean vector for $1/\sqrt{\mathrm{Var}[X] + \varepsilon}$. See Figure 91 for a visual example.

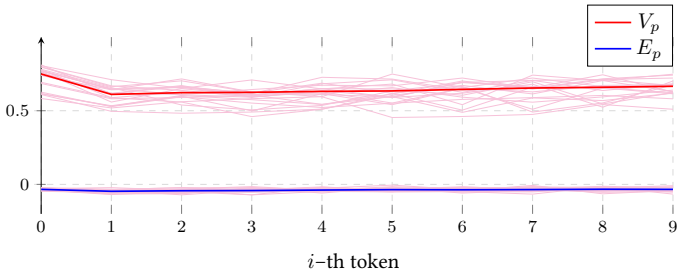


Figure 91: Example of distribution of $1/\sqrt{\mathrm{Var}[X] + \varepsilon}$ (top) and $\mathrm{E}[X]$ (bottom) for the first ten tokens in the very first Layer normaliztion on some inputs

This approach simplifies a lot the circuit, and it does not significantly affect the accuracy of the classifications, as will be shown in the experiments presented in Section 6.3. Let us now describe each section of Figure 90.

- *Section 1* – n Repeated ciphertexts coming from the previous layer.
- *Section 2* – the n MatVecRC procedures evaluate the first dense layer. The output rows are in Expanded shape.

- *Section 3* – a wrapping operation is executed in order to create a single ciphertext containing all the X_i needed by the following blocks.
- *Section 4* – the computation of (6.4) starts here. The value of $X_i - E_p$ is obtained by using the precomputed mean values for $\mathrm{E}[X]$. By taking advantage of SIMD computations, this is feasible in a single subtraction.
- *Section 5* – the previous result is then multiplied by $V_p\gamma$, which contains the precomputation of $\gamma/\sqrt{\mathrm{Var}[X] + \varepsilon}$.
- *Section 6* – the evaluation of the Layer normalization ends with the addition of the bias β.
- *Section 7* – The vectors contained in the ciphertext are unwrapped in n Expanded ciphertexts.

6.2.2.3 BertIntermediate

This layer is composed of two parts: a dense layer with 512 hidden units and the Gaussian Error Linear Unit (GELU) [HG23] activation function.

[HG23]: Hendrycks and Gimpel (2023), Gaussian Error Linear Units (GELUs)

The first matrix multiplication needs further investigation, since the size of the weight matrix is 128×512, which of course can not be encoded in a single plaintext. The approach is to split the matrix in four parts, so that they can be individually encoded. Remark that the input values are Repeated, meaning that the vector-matrix procedure to be used is MatVecER. The weight matrix, thus, must be split by columns and encoded by rows (Figure 92). This approach allows one to perform the

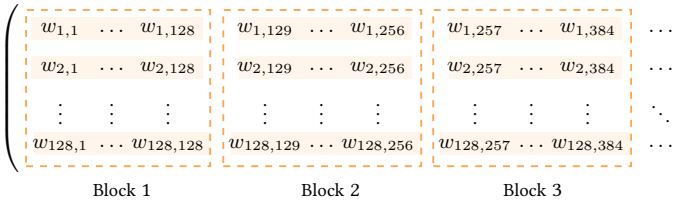


Figure 92: Splitting a 128×512 matrix into four 128×128 square matrices. The dashed squares represent the split over the columns, the highlight represents the Row-major encoding

actual vector-matrix multiplication by executing four times MatVecRC, each time with a different block. The first procedure returns the first 128 results, the second procedure the second 128 results, and so on.

All the obtained values will then be concatenated and fed as input to the $\mathrm{GELU}(x)$ activation function, which is defined as:

$$\mathrm{GELU}(x) = x \cdot \frac{1}{2}(1 + \mathrm{erf}(x/\sqrt{2})) \tag{6.5}$$

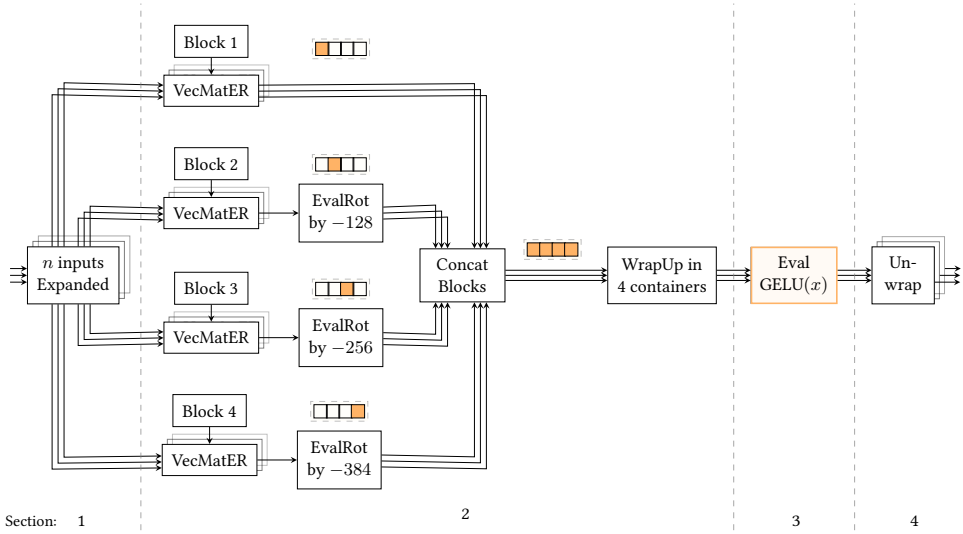


Figure 93: Circuit diagram illustrating the HE implementation of the BertIntermediate layer

where erf(x) is the so-called Error function erf(x), defined as:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t \tag{6.6}$$

In the following, all the sections from Figure 93 will be reviewed.

- *Section 1* – n Expanded ciphertexts received as input from the previous layer.
- *Section 2* – the MatVecER procedure is executed four times, one for each block of the weight matrix. The results obtained in the 128 positions are masked, so that the values can later be concatenated. Specifically, the first part calculates values ranging from 0 to 127, the second part computes values from 128 to 255, and so on. Rotations are needed because results are always located in the first 128 slots, hence they need to be shifted to be merged. By combining these parts together, we obtain the complete set of 512 resulting values for each vector.
- *Section 3* – the $4n$ ciphertexts need to be wrapped before evaluating GELU(x). Since each vector now contains 512 values, a single ciphertext will not be enough. Thus, we build four containers containing the wrapped up values. In general, $\lfloor n/32 \rfloor$ containers are required. Since results are already masked, the wrapping operation can be performed without further masking (hence, no levels consumption). Lastly, it is possible to evaluate GELU(x)

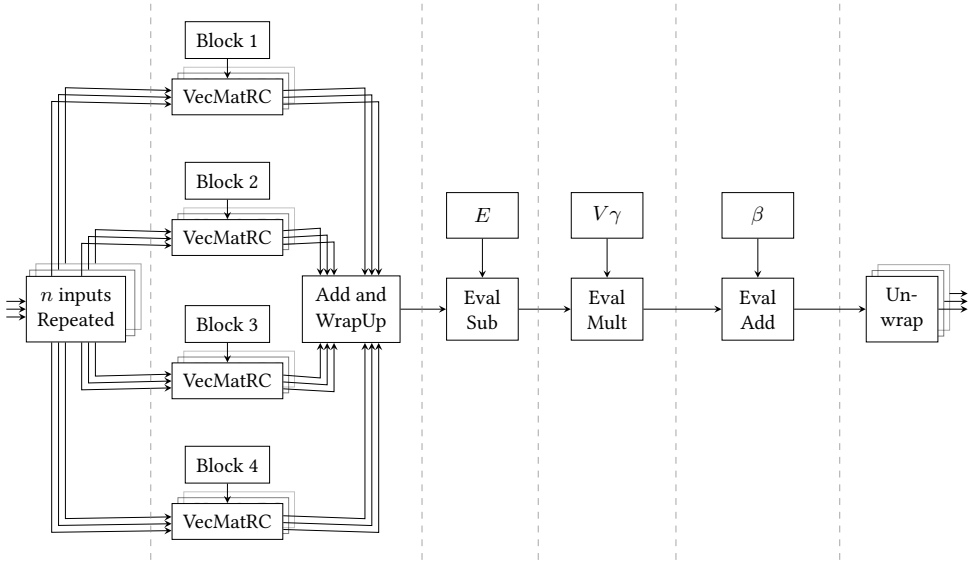


Figure 94: Circuit diagram illustrating the HE implementation of the BertOutput layer

on each container.

- *Section 4* – the results are unwrapped in $4n$ ciphertexts, four for each input. In particular, each block of 128 values in each ciphertext is stored in Repeated packing shape.

6.2.2.4 BertOutput

This is the last layer of the Transformer Encoder. It first reduces the number of features for each input from 512 back to 128 by evaluating a dense layer using a weight matrix of size 512×128. Like in the previous layer, performing this operation is not trivial because the weight matrix does not fit a single plaintext. Since the input ciphertexts are packed in Repeated mode, the corresponding matrix-vector multiplication to be performed is MatVecRC. The weight matrix is thus split along the rows and encoded by columns. Each column is split into four parts, and the results of these four multiplications must be added together in order to get the final results. In doing so, we obtain n ciphertexts in Expanded packing shape.

The subsequent part of the layer is a precomputed Layer Normalization, which has already been analyzed in Section 6.2.2.2 under the same input shapes (n Expanded ciphertexts).

Pooler. After two stacked encoders, $\text{BERT}_{\text{tiny}}$ implements a pooling layer, whose purpose is to provide a fixed-size representation of the entire input sequence. In particular, the Pooler layer operates on the final layer's hidden states and uses the output of the special [CLS] token, which is inserted at the beginning of the input sequence by the tokenizer. A fully connected layer is then evaluated on this vector, followed by the computation of the hyperbolic tangent $\tanh(x) = \sinh(x)/\cosh(x)$ activation function. It is possible to obtain a close approximation of this function by utilizing Chebyshev polynomials, even with a relatively low degree, since the observed interval of approximation required is $[-20, 20]$, which is pretty small.

Classifier. The output of the Pooler layer is a ciphertext in Repeated shape. The last operation is a MatVecRC procedure, in order to evaluate the last Classifier layer, composed of a single dense layer with two output neurons. This is the output of the circuit, which is sent back to the client.

6.3 Experiments

We now aim to assess the performance of the FHE circuit. All the experiments have been carried out according to the security level of 128 bits established by the Homomorphic Encryption Standards [Alb+18]. The reference dataset is the SST-2 dataset, which is composed of 67.3k training sentences extracted from movie reviews, each labeled with a positive or negative sentiment. The FHE circuit has been evaluated on a M1 Pro CPU, and it requires approximately 18GB of memory. The source code used for the experiments is freely available as an open-source repository[3].

[Alb+18]: M. Albrecht, Chase, H. Chen, Ding, et al. (2018), Homomorphic Encryption Security Standard

3: github.com/lorenzorovida/FHE-BERT-Tiny

Parameters. The parameters for the CKKS scheme have been chosen considering two constraints: $\lambda = 128$ bits of security and ciphertexts containing $s = 2^{14}$ values. The set of parameters is the following:

- Ring of dimension $N = 2^{16}$
- Ciphertext slots $s = N/4$
- Precision factor $\Delta = 55$ bits
- Moduli chain values $q_i = 52$ bits
- Circuit depth $d = 24$
- Ciphertext modulus $Q = 1767$ bits

[KPP22]: A. Kim, Papadimitriou, and Polyakov, *'Approximate Homomorphic Encryption with Reduced Approximation Error'* . In: CT-RSA '22

We refer the reader to Kim et al. [KPP22] for the details about the parameters. Since each embedding spans 128 dimensions, the circuit is able to handle at most $s/128 = 128$ tokens.

At first, the plain $\text{BERT}_{\text{tiny}}$ circuit has been evaluated on the whole training set, and the inputs of the non-linear functions have been experimentally observed. This step is required in order to determine the intervals in which Chebyshev polynomials will approximate the non-linear functions, and also to determine their degree, see Table 11.

Layer	Function	Interval	Degree
Encoder 1 – Self-Attention	$1/x$	$[2, 5000]$	119
Encoder 1 – Intermediate	$\text{GELU}(x)$	$[-14, 11]$	119
Encoder 2 – Self-Attention	$1/x$	$[3, 130000]$	200
Encoder 2 – Intermediate	$\text{GELU}(x)$	$[-18, 8]$	59
Pooler	$\tanh(x)$	$[-20, 20]$	300

Table 11: Chebyshev polynomials parameters of approximated non-linear functions in pre-trained HE model

Plain circuit. The chosen plain $\text{BERT}_{\text{tiny}}$ model[4] has been fine tuned on the sentiment analysis task over the SST-2 training set, and it achieved an accuracy of 0.837 on the validation set (we remark that the test set labels are not publicly available).

4: Available at huggingface.co/philschmid/tiny-bert-sst2-distilled

The first experiment aims to understand the performance loss introduced by using precomputed values for mean and variance in Layer normalization. As discussed in Section 6.2.2.2, we first obtained mean vectors for $\text{E}[X]$ and $1/\sqrt{\text{Var}[X] + \varepsilon}$ using the training set, then we used these values when evaluating the model on the validation set.

A performance drop from 0.837 to 0.815 has been observed, which is considered to be acceptable for our task, since the introduction of precomputed Layer normalization allows one to avoid the evaluation of four inverse square roots (two for each encoder), resulting in faster encrypted computations.

Encrypted circuit. The encrypted circuit is evaluated server-side, and the client requests for a classification as shown in Figure 83.

In our setting, the client has to tokenize the sentence and to encrypt the initial embeddings. The ciphertexts are then sent to the service provider, which we assume to be an honest-but-curious server. Then, the FHE circuit is evaluated and the output of the network will be a ciphertext containing two values. This is decrypted by the user, that is able to see the result of the classification.

When building a deep FHE circuit using a so-called *levelled* scheme[5], one must carefully consider the ciphertexts level growth, and choose when to perform the bootstrapping operations. In particular, this procedure must be performed when all the embeddings are wrapped in a single ciphertext and ideally when values are close to the interval $[-1, 1]$, in order to obtain more accurate results.

5: A levelled scheme allows for a fixed number of multiplications (in our case d) before requiring a bootstrapping.

Figure 95 presents how the level of the ciphertexts grows during the evaluation of the circuit, and where we decided to perform a bootstrapping.

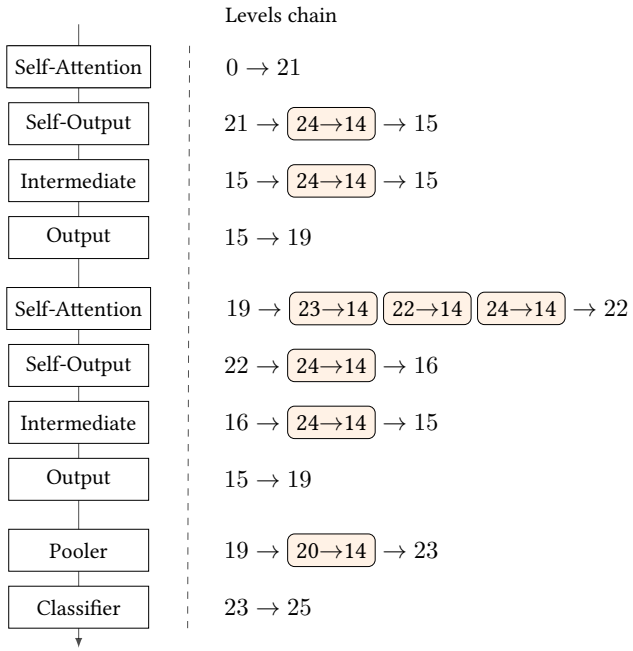


Figure 95: Ciphertexts levels progress during the evaluation of the circuit. An orange block means that a bootstrapping operation is performed.

The operation is not always performed at the last level (24) because it could happen that, at that level:

- The embeddings are not wrapped in a single ciphertext.
- There is an evaluation of a Chebyshev polynomial, which can not be discontinued by a bootstrapping operation because of the recursive nature of the Paterson-Stockmeyer [PS73] polynomial evaluation algorithm.
- The values of the ciphertext are much larger than 1.

If one of these cases occur, bootstrapping must be performed beforehand.

[PS73]: Paterson and Stockmeyer (1973), *'On the number of nonscalar multiplications necessary to evaluate polynomials'* . In: SIAM Journal on Computing

To begin with, we first want to give an interpretation of the result given by the FHE circuit. The first aspect that is

analyzed is indeed the error introduced by the approximate computations, and how it impacts the final classification. We remark that the classification is performed by finding the most active between the two final neurons (i.e., which one contains the largest value). Therefore, a classification is considered to be correct unless the approximations cause an inversion of the order between the two output values. A visual representation of the impact of approximations is given in Figure 96. More

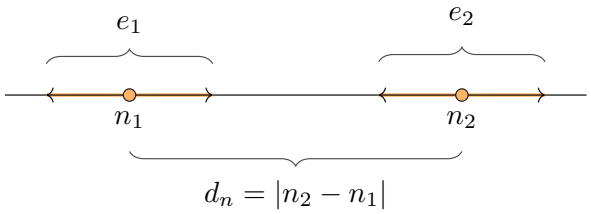


Figure 96: Example of two output values n_1, n_2 and errors e_1, e_2 added by the approximate FHE circuit.

formally, a classification is incorrect if $d_n < (e_1 - e_2)/2$. In that case, n_1 might become larger than n_2 (or vice versa). To summarize, the probability to obtain a correct classification depends on:

- The amount of error introduced by approximations ($(e_1 - e_2)/2$).
- The distance between the values of the two output neurons (d_n).

The first aspect can be controlled by carefully selecting the parameters Δ and q_i of the CKKS scheme (defined in Section 6.3) and by the approximations of the non-linear functions (see Table 12). The second aspect, on the other hand, is an intrinsic characteristic of the model.

Before presenting the results, we want to emphasize that small approximations are not always indicative of correct classification, and vice versa. For example, by considering the sentence "*it 's a work by an artist so in control of both his medium and his message that he can improvise like a jazzman.*", the output values have a distance $d_n \approx 0.018$. This could happen because the features extracted by $\text{BERT}_{\text{tiny}}$ are not enough for the classifier to well distinguish between the two sentiments. In this case, even a small error could cause a wrong classification. On the other hand, the output values for the sentence "*the best film about baseball to hit theaters since field of dreams.*" have a distance $d_n \approx 3.897$, meaning that even large errors could result in correct classification. We define the function used to evaluate the overall error between the expected (plain) and

obtained (FHE) values as:

$$\text{error}(\boldsymbol{v}, \boldsymbol{v}') = \sum_{i=1}^{\text{len}(\boldsymbol{v})} \left(\left| \frac{v_i - u_i}{\max(\boldsymbol{v})} \right| \right) \cdot \|\boldsymbol{v}\|^{-1} \tag{6.7}$$

where $\boldsymbol{v}$ represents the expected vector, and $\boldsymbol{v}'$ the obtained one. We present in Figure 97 the distances between the output values (d_n) computed in plain on the entire validation set, and the errors obtained by the FHE circuit, evaluated on the same set. We observed that the main cause of errors is the Chebyshev

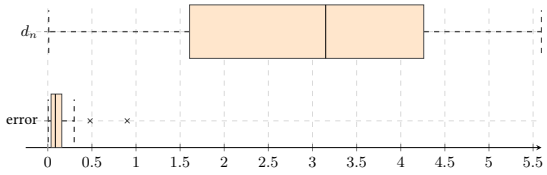


Figure 97: Errors, relative to the FHE circuit, and the distance between the output values (d_n) obtained from evaluating the validation set.

approximation of the $1/x$ function, evaluated in the Self-Attention layer of the second encoder, since its approximation interval is very large, as shown in Table 11. Nevertheless, errors are usually below the output values, meaning that the FHE circuit should achieve a similar performance to the plain one. To complete the evaluation, we present the accuracy obtained by the FHE circuit in Table 12. The loss in accuracy,

Model	Accuracy	Loss
$\text{BERT}_{\text{tiny}}$	0.837	(*reference*)
$\text{BERT}_{\text{tiny}}$ with precomputed LN	0.815	0.022
FHE-$\text{BERT}_{\text{tiny}}$	0.790	0.047

Table 12: Accuracy for the three considered models, on the SST-2 validation set. LN stands for Layer Normalization

with respect to the $\text{BERT}_{\text{tiny}}$ model with precomputed Layer Normalization, is -0.025.

Then, we concentrate on computational times; in particular, we want to highlight the correlation between the execution time and the number of tokens n.

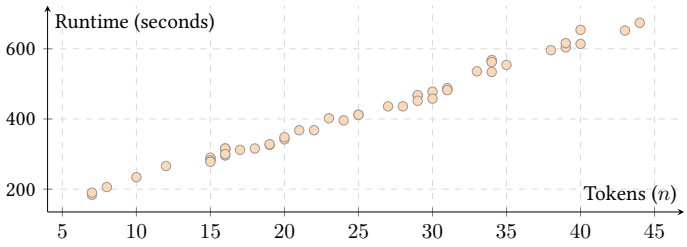


Figure 98: Linear correlation between the FHE circuit runtime and the number of tokens

As confirmed by Figure 98, there is a strong linear correlation between computational time and the number of tokens.

6.4 Conclusion

In this chapter, we proposed a circuit which implements $\text{BERT}_{\text{tiny}}$ [BDR21] based on FHE primitives, precisely on the RNS-CKKS scheme [KPP22]. In particular, the setting is composed of an honest-but-curious server that offers the language model as a service to a client, which has to tokenize and create token embeddings of a sentence. These vectors are then encrypted and sent to the service provider, that computes the Transformer encoder layers, a pooling layer and a classification layer in a privacy-preserving environment. The circuit has been implemented using the OpenFHE library on a M1 Pro CPU, and it requires 18GB of RAM. The source code of the circuit is available at our open-source repository. Results are promising, although this proposal is a baseline from which various improvements can be carried out. First of all, it would be interesting to scale the architecture to support larger BERT models, but also to test this circuit on more accurate polynomial approximations of activation functions, to see how the results change. In HE literature it is possible to find references to HE-friendly networks, in the sense that they are not based on non-linear functions. Considering a wider context, it would be also interesting to study a class of neural networks which output well-separated logits, so that the same results could be obtained using less accurate HE computations. This would translate in faster and lighter calculations.

[BDR21]: Bhargava, Drozd, and Rogers (2021), *'Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics'* . In: Proceedings of the Second Workshop on Insights from Negative Results in NLP

[KPP22]: A. Kim, Papadimitriou, and Polyakov, *'Approximate Homomorphic Encryption with Reduced Approximation Error'* . In: CT-RSA '22

Efficient sorting algorithms for encrypted numbers

7

This chapter is an extended/revised version of the joint work 'Lightweight sorting in approximate Homomorphic Encryption'*, with S. Basile and A. Leporati, under review in IACR Communications in Cryptology*

Abstract: Sorting encrypted values is an open research problem that plays a crucial role in the broader objective of providing efficient and practical privacy-preserving online services. The current state of the art work by Mazzone, Everts, Hahn and Peter (USENIX Security '25) proposes efficient algorithms for ranking, indexing and sorting based on the CKKS scheme, which deviates from the compare-and-swap paradigm, typically used by sorting networks, using a permutation-based approach. This allows to build shallow sorting circuits in a very simple way. In this work, we follow up their work and explore different approaches to approximate the nonlinear functions required by the circuit. We propose simpler solutions that allow for faster computations and smaller memory requirements. In particular, we drastically reduce the upper bound on the depth of the circuits from 65 to 20, making our circuits usable in relatively small rings such as $N = 2^{16}$ – even for sorting values while preserving up to three decimal places. Furthermore, we propose an implementation of a swap-based bitonic network that is not based on approximations of the sgn(x) function, which scales linearly with the number of values, useful when the number of available slots is small.

7.1 Introduction

Using FHE, it is possible to perform computations with the same computational power as Turing machines [Gol+13], but over encrypted data; although there is still a huge gap between theory and practice. For example, evaluating a conditional statement represents one of the main challenges in FHE, since it implies the evaluation of a nonlinear function. Classical sorting algorithms are heavily based on conditional statements, therefore evaluating them on encrypted values is far from being an easy task. Nevertheless, the need for efficient and quick sorting algorithms is high

[Gol+13]: Goldwasser, Kalai, Popa, Vaikuntanathan, et al., *'How to Run Turing Machines on Encrypted Data'* . In: CRYPTO '13

since many protocols, such as encrypted machine learning and private information retrieval, heavily rely on sorting results as a subroutine. The goal of this chapter is thus to propose lightweight algorithms to sort encrypted values based on CKKS [CKKS17], which is an approximate FHE scheme.

7.1.1 Related works

Sorting encrypted values. One of the first attempts to solve the problem of sorting a set of encrypted values was published in 2013 by Chatterjee, Kaushal and Sengupta [CKS13]. A couple of years later, in 2015, Emmadi *et al.* [Emm+15] proposed a list of sorting algorithms based on FHE primitives. Nonetheless, these preliminary works were mostly theoretical and presented methods to sort a small amount of values (no more than 64 values) in a few hours, typically by using Boolean circuits.

A few years later, following the *scheme-switching* approach, Lu *et al.* [Lu+21] proposed an algorithm based on bitonic sorting that exploited the scheme-switching capability to take advantage of look-up-tables for the evaluation of the comparison operation $a < b$ from CKKS to TFHE. Their approach was able to sort 32 values in less than 10 minutes using an Intel Xeon Platinum 8269CY CPU, in single-thread. Nevertheless, the scheme-switching capability is very costly and does not scale much with the number of values, since each value requires a look-up-table evaluation.

In 2021, Hong *et al.* [Hon+21] proposed a so-called k-way sorting network based entirely on the CKKS scheme in which comparisons are performed using polynomial approximations of the $\mathrm{sgn}(x)$ function. According to their estimations, sorting 512 values required roughly 150 minutes using an AMD Ryzen 9 3950X CPU, and more than 23 GB of RAM memory. We will refer to this approach as *network-based sorting*. This top-k approach was then improved and extended by Cong, Geelen, Kang and Park [Con+25] under the TFHE scheme [CGGI20], with applications to k-NN classifiers.

We will consider the state of the art work by Mazzone, Everts, Hahn and Peter [MEHP25], where efficient algorithms for sorting based on CKKS are presented. In particular, they propose procedures that are not based on the swap operations implemented in the previous work based on sorting networks. Their sorting algorithm can be divided into three steps: (i) find the indexing of the input vector $\boldsymbol{v}$ (i.e., the index of each

[CKS13]: Chatterjee, Kaushal, and Sengupta, *'Accelerating Sorting of Fully Homomorphic Encrypted Data'* . In: INDOCRYPT '13

[Emm+15]: Emmadi, Gauravaram, Narumanchi, and Syed, *'Updates on Sorting of Fully Homomorphic Encrypted Data'* . In: ICCCRI '15

[Lu+21]: Lu, Z. Huang, C. Hong, Ma, et al., *'PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption'* . In: S&P '21

[Hon+21]: S. Hong, S. Kim, Choi, Y. Lee, et al. (2021), *'Efficient Sorting of Homomorphic Encrypted Data With k-Way Sorting Network'* . In: IEEE Trans. on Information Forensics and Security

[Con+25]: Cong, Geelen, Kang, and J. Park, *'Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification'* . In: SAC '24

[MEHP25]: Mazzone, Everts, Hahn, and Peter, *'Efficient Ranking, Order Statistics, and Sorting under CKKS'* . In: USENIX '25

value v_i in the sorted vector $\text{sort}(\boldsymbol{v})$), (ii) given the indexing, build the corresponding permutation matrix P, and (iii) sort $\boldsymbol{v}$ by evaluating a matrix-vector multiplication. Although very practical for small sets of numbers, this approach requires storing $\Theta(n^2)$ slots, where n is the number of values to be sorted, indeed the authors propose to split the input in *chunks* of data when the number of values is too large, i.e., when $n > 256$. One of the main advantages of this approach is that the circuit depth, in terms of multiplications, is very small compared to sorting networks. We will refer to the their approach as *permutation-based sorting*.

Approximate comparison operations. A part of our work includes an efficient sorting network which heavily relies on comparing encrypted values. In literature, comparisons are typically performed using (polynomial approximations of) the $\text{sgn}(x)$ function on the difference between two values. Usually, in the CKKS scheme, nonlinear functions are evaluated via polynomial approximations. Cheon *et al.* [Che+19] proposed, in 2019, some approximations of the $\text{sgn}(x)$ function by defining compositions of polynomials. Plus, they also defined an approximation of the $\min(a, b)$ function[1] based on the fact that:

$$\min(a, b) := (a + b)/2 - \sqrt{(a - b)^2}/2.$$

In this case, the square root was evaluated using an iterative method [Wil51].

[Che+19]: Cheon, D. Kim, D. Kim, H. H. Lee, et al., *'Numerical Method for Comparison on Homomorphically Encrypted Numbers'* . In: ASIACRYPT 19

1: Notice that the $\max$ function can be trivially obtained from $\min$ as $\max(a, b) = a + b - \min(a, b)$

Cheon, Kim and Kim [CKK20] observed that $(\sqrt{x})^2 = x \cdot \text{sgn}(x)$, enabling to use approximations of the $\text{sgn}(x)$ function to compute the minimum and maximum between two elements as $\min(a, b) = (a + b)/2 - \sqrt{(a - b)^2}/2$. Nevertheless, these approaches required a large multiplicative depth (i.e., the number of multiplications performed on a ciphertext) to ensure a decent amount of precision. Lee *et al.* [Lee+22c] proposed an approach based on composite Minimax polynomials which showed smaller execution time and a smaller multiplicative depth. We remark that [Lee+22c] proposed to evaluate the $\max(a, b)$ and $\min(a, b)$ functions, along with the Rectified-Linear Unit (ReLU) function, as

[CKK20]: Cheon, D. Kim, and D. Kim, *'Efficient Homomorphic Comparison Methods with Optimal Complexity'* . In: ASIACRYPT '20

[Lee+22c]: E. Lee, J.-W. Lee, No, and Y.-S. Kim (2022), *'Minimax Approximation of Sign Function by Composite Polynomial for Homomorphic Comparison'* . In: IEEE Transactions on Dependable and Secure Computing

$$\begin{aligned} \min(a, b) &\approx \frac{(a + b) - (a - b) \cdot p(a - b)}{2} \quad \text{and} \\ \text{ReLU}(x) &\approx \frac{x + x \cdot p(x)}{2}, \end{aligned} \tag{7.1}$$

where $p(x)$ is some polynomial approximation of the $\text{sgn}(x)$ function. We highlight that, with this approach, $\max(a, b)$,

$\min(a, b)$ and $\text{ReLU}(x)$ are computed using $p(x)$, which is an approximation of a non continuous step function. Lastly, another line of works propose to perform the evaluation of non-linear functions via the *programmable bootstrapping* approach of the TFHE scheme. Liu, Micciancio and Polyakov [LMP23] proposes a large-precision evaluation of the $\text{sgn}(x)$ function in which the precision scales logarithmically with the ciphertext modulus, making it more usable in practice.

[LMP23]: Z. Liu, Micciancio, and Polyakov, *'Large-Precision Homomorphic Sign Evaluation Using FHEW/TFHE Bootstrapping'* . In: ASIACRYPT '22

Comparison with [Kim+25]. In a concurrent work, Kim, Ünay, Yilmazer-Metin and Lee [Kim+25] proposed solutions to sorting which can be considered close to our approach, especially considering the usage of the $\text{sinc}(x)$ function. We believe that some of [Kim+25] ideas could be applied to our permutation-based sorting framework, especially when sorting large amount of values. Nevertheless, we do not consider our works as colliding with each other, as their approach is based on the $N = 2^{17}$ ring and polynomial compositions are used to approximate the signum function – in toto circuits in [Kim+25] can require up to 60 multiplicative levels. On the other hand, our proposal is mainly focused on lightweight computations, and we believe that our framework can be improved with some of [Kim+25] techniques. As it will be shown, our circuits can be instantiated using the smaller $N = 2^{16}$ ring and Chebyshev polynomials, requiring at most 20 levels in the same setup.

[Kim+25]: S. Kim, Ünay, Yilmazer-Metin, and H. T. Lee (2025), Optimized Rank Sort for Encrypted Real Numbers (Cryptology ePrint Archive, Paper 2025/1170)

7.1.2 Our contribution

We split our proposal in two parts:

1. We follow-up [MEHP25] and explore different and more efficient approaches to approximate the required non-linear functions in their permutation-based sorting algorithm. We will show how to reduce the depth of the circuit, reducing the upper bound from 65 to 20 levels, making the sorting faster and with a smaller memory footprint. In order to keep the circuit efficient, we will not use rings larger than $N = 2^{16}$, keeping $\lambda \geq 128$ bits of (classical) security.
   In particular, we propose to simplify their framework by removing non essential approximations of step functions. We will use the k-scaled sigmoid function in the first ranking procedure, and the $\text{sinc}(x)$ function to perform the equality check required to construct the permutation matrix P and the tie-offset correction.

The sinc(x) function can be accurately approximated without requiring many multiplicative levels, as it is continuous, yet it allows to obtain the same performance as the more complex counterparts based on sgn(x) – under some assumptions, i.e., the input of sinc(x) must be as close as possible to $\mathbb{Z}$. Both will be approximated using Chebyshev polynomials. The number of required slots for the ciphertexts to contain the input is quadratic $\Theta(n^2)$, where n is the number of values to sort.

2. We construct an efficient sorting network using bitonic sorting that, differently from [Hon+21], is based on a definition of the $\min(a, b)$ function that does not depend on the sgn$(a - b)$ function. This allows us to reduce the depth of the circuit and to obtain more accurate computations. We aim to approximate the $\min(a, b)$ and the $\max(a, b)$ functions using an approximation of the $\max(0, x)$ function (i.e., the ReLU function):

$$\begin{aligned} \min(a, b) &:= a - \max(0, a - b) \\ &= a - \text{ReLU}(a - b). \end{aligned} \tag{7.2}$$

The advantage of this approach is that $\max(0, a - b)$ is easier to approximate, even with Chebyshev polynomials, with respect to the polynomial $p(x)$ used in [Lee+22a; Lee+22c], since it is continuous and does not contain steps. We show that, with this approach, it is possible to obtain large level of precision bits α with a small multiplicative depth (e.g., $\alpha > 10$ with depth $d = 9$). By performing swaps using (7.2), computational time is orders of magnitude smaller than previous works [Hon+21; Hua+22; Lu+21], and better scales with the number of inputs. The number of required slots for the ciphertexts to contain the input is linear $\Theta(n)$ and can become a valid alternative in applications where the number of available slots is not as large as $\Theta(n^2)$ – namely the slots required by the more efficient permutation-based approach.

We show quantitative results of our approach in several experiments, which are reproducible using the OpenFHE library with the C++ code present in our open-source repository[2]. The repository also contains some notebooks that implement the algorithms as a plain circuits in Python.

2: github.com/lorenzorovida/lightweight-sorting-fhe

7.2 Preliminaries

In what follows, if not stated otherwise, lowercase letters are used to indicate numbers (e.g., a), bold-italic lowercase letters to indicate vectors (e.g., $\boldsymbol{v}$), and uppercase letters are used to indicate matrices (e.g., A). We extend the vectors notation for single numbers, e.g., $\mathbf{1}^n = (1, 1, \ldots, 1)$ or scalars $\boldsymbol{i}^n = (i, i, \ldots, i)$. If not specified, the dimension should be clear by the context.

Given a vector $\boldsymbol{v}$, we define $\boldsymbol{v}_{i:j}$ as the subvector containing the elements from index i to j, namely $\boldsymbol{v}_{i:j} = (v_i, v_{i+1}, \ldots, v_j)$.

Vectors are considered as column vectors, the i-th column of a matrix A is referred to as $\boldsymbol{a}_i$ and the i-th element of a vector $\boldsymbol{v}$ as v_i. If not specified, we assume $\log(x) = \log_2(x)$. We denote the flattened outer product of two vectors using the symbol $\boldsymbol{a} \otimes \boldsymbol{b}$: in order to obtain a vector out of the resulting matrix, the result of a outer product is flattened row-wise as a vector, therefore we denote $\otimes : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n^2}$. For instance, $[x, y] \otimes [w, z] = [xw, xz, yw, yz]$. This trick allows us to easily create n repetitions of a vector $\boldsymbol{v}$ simply as $\mathbf{1} \otimes \boldsymbol{v}$, or to repeat each element of a vector n times as $\boldsymbol{v} \otimes \mathbf{1}$.

7.2.1 Some oblivious sorting algorithms

Sorting algorithms in the encrypted realm represents a challenge since most of the common approaches to sort values are *data-dependent*. Indeed, standard algorithms using clear values allow to define worst, average and best cases. In our setting, however, since values are encrypted, standard algorithms fail to be valid alternatives as it is not possible to change the flux of the execution based on the input data. We now present two of the standard approaches used in literature to sort encrypted values: permutation-based and network-based. Note that we will now present the algorithms in plain form, abstracting from any particular encryption scheme.

Permutation-based sorting. The goal of permutation-based sorting is to sort a vector $\boldsymbol{v} \in \mathbb{R}^n$ by finding the permutation matrix $P \in \{0, 1\}^{n \times n}$ such that $P \cdot \boldsymbol{v}$ is correctly sorted. The main objective is thus the construction of the P matrix. The main advantage of this approach is that the depth of the circuit is somewhat small, with respect to sorting networks. In [MEHP25], they propose to find P in two steps:

1. Finding the indexing of the vector $\boldsymbol{v} \in \mathbb{R}^n$. Namely, construct a vector $\boldsymbol{s} \in \mathbb{Z}^n$ such that, if $\boldsymbol{v}$ was sorted, the i-th element of $\boldsymbol{v}$ would be in position s_i, starting from 1. For example, given $\boldsymbol{v} = (4, 7, 1, 8)$ we get $\boldsymbol{s} = (2, 3, 1, 4)$. This operation can be achieved by comparing each element v_i with all the others. For example, the first element s_1 is given by how many elements in $\boldsymbol{v}$ are smaller than v_1. In order to do that, the $\text{sgn}(x)$ function is used on all the possible combinations $v_i - v_j$, where:

$$\text{sgn}(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

Following the previous example, the first element s_1 is evaluated as:

$$s_1 = 0.5 + \text{sgn}(v_1 - v_1) + \text{sgn}(v_1 - v_2) + \text{sgn}(v_1 - v_3) + \text{sgn}(v_1 - v_4).$$

In general, we have that

$$s_i = 0.5 + \sum_{j=1}^{n} \text{sgn}(v_i - v_j).$$

2. The second step is, given the indexing vector $\boldsymbol{s} \in \mathbb{Z}^n$, to construct the corresponding permutation matrix $P \in \{0, 1\}^{n \times n}$. Each i-th column $\boldsymbol{p}_i \in \{0, 1\}^n$ of P is constructed using an equality check. We define the $\text{equal}(x)$ function as:

$$\text{equal}(x) := \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ otherwise} \end{cases} .$$

By abuse of notation we allow the function to be applied on vectors by evaluating the corresponding function independently on each entry. Finally, the i-th column $\boldsymbol{p}_i$ of the permutation matrix P can be found as

$$\boldsymbol{p}_i := \text{equal}(\boldsymbol{s} - \boldsymbol{i}).$$

with $0 < i \leq n$. Following the example, the first column $\boldsymbol{p}_1$ of P is computed as:

$$\boldsymbol{p}_1 := \text{equal}\,(\boldsymbol{s} - \boldsymbol{1}) = \text{equal}\,((2, 3, 1, 4) - (1, 1, 1, 1)) = (0, 0, 1, 0)$$

This approach well fits the CKKS scheme, since the latter allows to parallelize the many operations with SIMD compu-

tations. As a result, this "horizontal" approach allows to avoid bootstrapping as the depth of the circuit can be roughly defined by the sum of the depth required by the approximations of $\text{sgn}(x)$ and $\text{equal}(x)$.

Tie-correction offset. One may notice that this method fails to sort a vector which contains repeated values, e.g. $\boldsymbol{v} = (4, 7, 1, 1)$. An effective solution proposed in [MEHP25], is to intervene in the indexing vector, which, using $\boldsymbol{v}$, would be constructed as $\boldsymbol{s} = (3, 4, 1.5, 1.5) \notin \mathbb{Z}^n$. One solution is to apply a *tie-correction offset*. In particular, we require to construct an offset vector that corrects the partial indexing. Following the example, such a vector would be $\boldsymbol{o} = (0, 0, -0.5, 0.5) \in \mathbb{R}^n$. Therefore, the final indexing would be given by the sum $\boldsymbol{s} + \boldsymbol{o} \in \mathbb{Z}^n$. The offset vector $\boldsymbol{o} \in \mathbb{R}^n$ can be obtained as follows:

1. Count how many elements are repeated, and how many times. Formally, for each element $v_i \in \boldsymbol{v}$, count how many elements v_j are equal to v_i, with $0 < i, j \leq n$. We build the "repetitions" vector $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ that contains such information, where each $0 < r_i \leq n$, as:

$$r_i := \sum_{j=1}^{n} \text{equal}(v_i - v_j).$$

2. Define a vector $\boldsymbol{k}$ which contains information about the partial repetitions. In practice, each k_i is equal to the number of elements in $\boldsymbol{v}_{1:i}$ that are equal to v_i. Single elements will have sum 1, while each repeated value will be assigned a value from 1 to x, where x is the number of repetitions.

$$k_i := \sum_{j=1}^{i} \text{equal}(v_i - v_j).$$

3. Compute $\boldsymbol{o} := \big(\boldsymbol{k} - (0.5 \cdot \boldsymbol{r}) - \mathbf{0.5}\big) \in \mathbb{R}^n$ to obtain the final tie-correction offset $\boldsymbol{o} \in \mathbb{R}^n$. Following the example, given $\boldsymbol{v} := (4, 7, 1, 1) \in \mathbb{R}^n$ and the corresponding indexing $\boldsymbol{s} := (3, 4, 1.5, 1.5) \notin \mathbb{Z}^n$:

$$\begin{aligned} \boldsymbol{o} := \boldsymbol{k} - (0.5 \cdot \boldsymbol{r}) - \mathbf{0.5} &= (1, 1, 1, 2) - (0.5, 0.5, 1, 1) - \mathbf{0.5} \\ &= (0, 0, -0.5, 0.5) \in \mathbb{R}^n \end{aligned}$$

which is precisely the offset we were looking for. Refer to [MEHP25] for a proof of correctness and more details

about this procedure.

Network-based sorting. In sorting networks the only operation that depends on the input data is the *compare-and-swap* operation, performed if two elements are not in the correct order. The latter is defined as follows:

$$\textbf{if } x > y \textbf{ then } \text{swap}(x, y),$$

and it can be hardware implemented as shown in Figure 99. In the context of sorting networks, this operation is usually

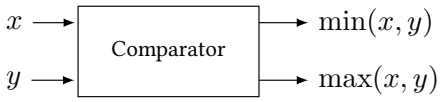


Figure 99: Compare-and-swap in hardware

represented as illustrated in Figure 100. In recent FHE liter-

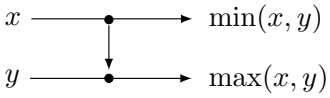


Figure 100: Compare-and-swap In a sorting network

ature [Hon+21], this operation has been implemented using approximations of the $\text{sgn}(x)$ function to mimic the comparison $a > b$.

Bitonic sorting. We propose to build a sorting network using the *bitonic sorter* algorithm [Akl11], which sorts a power of two amount of values. In particular, given n inputs (with n being a power of two), the number of layers of the corresponding sorting network will be equal to:

$$\frac{\log(n)(\log(n)+1)}{2} \tag{7.3}$$

Informally, the bitonic sorting algorithm builds a sorting network that, at the beginning, sorts each 2-consecutive elements, then each 4-consecutive elements, and so on. Let us introduce two definitions that will be useful later.

Definition 65 (*k-bitonic sequence*)

A k-bitonic sequence *is a sequence of* n *values such that* $x_1 \leq x_2 \leq \cdots \leq x_k \geq x_{k+1} \geq \cdots \geq x_n$*, with* $k \in [1, n]$*. For convenience, a non-bitonic sequence is considered a 1-bitonic sequence, and an* n*-bitonic sequence is a sorted sequence.*

Definition 66 (*k-bitonic block*)

A k-th bitonic block, *with $k \in \mathbb{Z}^+$, is a sequence made of $\log(k) + 1$ sorting layers, such that, given as input a k-bitonic sequence, returns as output a $2k$-bitonic sequence.*

Note that, in general, these definitions can be applied to any subsequence. For example, the 2-bitonic sequence $[1, 2, 3, 0, 4, 9, 9, 1]$, given as input to a 2-bitonic block, produces a 4-bitonic sequence $[0, 1, 2, 3, 9, 9, 4, 1]$. A network that sorts n values will be composed of $\log(n)$ bitonic blocks. For instance, Figure 101 illustrates a network that sorts $n = 8$ input values.

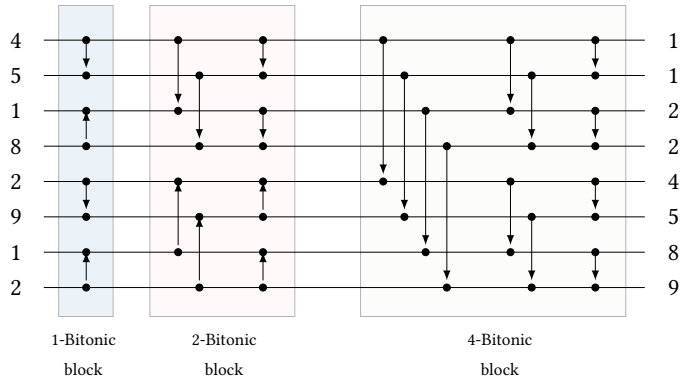


Figure 101: A bitonic sorting network composed of 6 layers that sorts $n = 8$ input values. The direction of the arrows defines the order of the comparisons

In this example, eight values enter the network from the left, move across the eight horizontal wires, and exit from the right. The network sorts them, placing the largest number at the bottom. We will show how to evaluate such a network with only 6 comparison operations (7.3). As opposed to the permutation-based approach, these networks tend to require deep circuits to be evaluated. On the other hand, they do not require to repeat the input n times in the ciphertext.

We skip the basic definitions of CKKS and Chebyshev polynomials, and we refer to Section 3.3 and Section 5.1.3.

7.3 Methodology

We will now describe our concrete CKKS-based approach to construct more efficient permutation-based sorting algorithms and bitonic sorting networks.

7.3.1 Permutation-based sorting

This approach relies on the computation of two nonlinear functions only (three, if accepting repeated elements). Note that our goal is not to prove the correctness of the procedures – whose proofs can be found in [MEHP25] – but to propose concrete approaches to make them computationally lighter. Let us define $\boldsymbol{c} \in \mathbb{R}^n$ as the (encrypted) input vector[3], where n is without loss of generality considered a power of two for efficiency reasons.

3: For the sake of clarity, we will for now make no difference between clear and encrypted vectors since every operation in the described procedure can be implemented via additions and multiplications.

Indexing based on SIMD. The first function, $\text{sgn}(x)$ is used to count, for every $c_i \in \boldsymbol{c}$, how many elements in the input vector are smaller than c_i. The SIMD property of CKKS allows to compute this very efficiently in a single comparison by repeating the input vector and using $\Theta(n^2)$ slots of the ciphertext. We require the input to be encoded in two different ways: *repeated* encoding (the whole vector $\boldsymbol{c}$ is repeated n times) and *expanded* encoding (each element c_i is repeated n times). Given these encodings, which can be easily obtained as:

$$\boldsymbol{c}_{\text{expanded}} := \boldsymbol{c} \otimes \mathbf{1} \in \mathbb{R}^{n^2} \quad \text{and} \quad \boldsymbol{c}_{\text{repeated}} := \mathbf{1} \otimes \boldsymbol{c} \in \mathbb{R}^{n^2},$$

we start by performing a subtraction between them. We assume both encodings to be immediately available. If not, one can compute them, starting from $\boldsymbol{c}$, at the cost of 1 multiplication and $\log(n)$ additions and rotations. A visual example is given in Figure 102, considering $n = 4$ and the input vector $\boldsymbol{c} := (4, 7, 1, 8)$.

$\boldsymbol{c}_{\text{repeated}}$:	4	7	1	8	4	7	1	8	4	7	1	8	4	7	1	8
$\boldsymbol{c}_{\text{expanded}}$:	4	4	4	4	7	7	7	7	1	1	1	1	8	8	8	8
$\boldsymbol{c}_{\triangle}$:	0	3	−3	4	−3	0	−6	1	3	6	0	7	−4	−1	−7	0

Figure 102: Computing $\boldsymbol{c}_{\triangle}$ from the two encodings of the input ciphertext $\boldsymbol{c}$ to be sorted

At this point, the idea is to count the occurrence of positive elements in each n-sized subvector. In order to do that, the $\text{sgn}(x)$ function is applied to the whole $\boldsymbol{c}_{\triangle}$ ciphertext. By the means of SIMD, this operation is performed in a single step. We will later present our approach to evaluate such a function in CKKS. Then, in order to get the sum of the values in positions $k \cdot n$ (which refer to the k-th element in $\boldsymbol{c}$) we rotate and sum elements in positions $k \cdot n$, with $0 \leq k < n$ integer (this procedure is denoted in Figure 103 by the $\sum$

symbol) and we increase the results by 0.5 to correct the bias introduced on elements whose subtraction lead to 0.

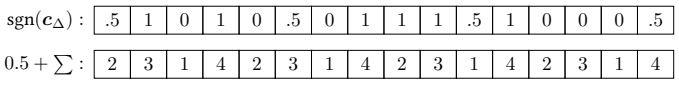


Figure 103: Given $\boldsymbol{c}_\Delta$, the (repeated) indexing of $\boldsymbol{c}$ is given by performing a rotate and sum (indicated with the $\sum$ symbol) over sgn($\boldsymbol{c}_\Delta$), plus $\mathbf{0.5}$

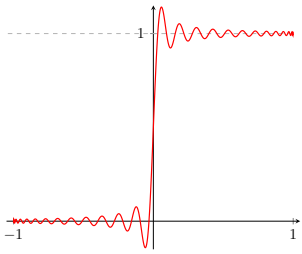
Notice that this procedure returns the repeated indexing vector (following the example, $\mathbf{1} \otimes (2, 3, 1, 4)$) and it is pretty quick, apart from its main bottleneck, i.e., the sgn(x) function, since it has to be approximated with high degree polynomials. In order to do that, it is possible to follow two directions:

- Using the approach firstly introduced in [CKK20], and followed in [MEHP25], where the authors propose to approximate sgn(x) using compositions of known polynomials $f(\cdot)$ and $g(\cdot)$. This path leads to smaller runtime and to more accurate computations, although it has the drawback of requiring a large multiplicative depth.
- Using Chebyshev polynomials to directly approximate sgn(x). These have the advantage of giving decent approximations using less levels, although they require more computational time to obtain a large amount of precision.

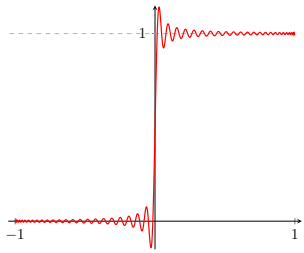
Since our goal is to present low-depth implementations (i.e., we avoid using large CKKS rings such as $N = 2^{17}$), we stick to Chebyshev polynomials. Approximating the raw sgn(x) function, though, is not a good choice, as the resulting polynomial will suffer by a strong Gibbs phenomenon, namely large oscillations near the "step" occurring in $x = 0$, as shown in Figure 104. We thus propose to define a priori some value $\delta > 0$ that defines the minimum distance between any pair of elements in the vector to be sorted – ignoring repeated values. In the experiments presented in [MEHP25], this is implicitly set to $\delta = 0.01$, although they conveniently report sets of parameters to achieve any desired level of precision in their open-source repository. We define the k-scaled sigmoid function as

$$\text{sigmoid}_k(x) := \frac{1}{1 + e^{-kx}}. \tag{7.4}$$

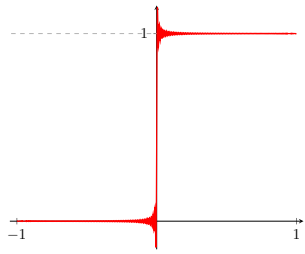
Notice that this function acts as a "smooth" sgn(x) function, where the smoothness parameter is given by $k \in \mathbb{Z}^+$. Our strategy is to empirically find some value k such that the sigmoid well approximates sgn(x), up to some negligible error, in $[-1, -\delta] \cup [\delta, 1]$. As an example, we fix $\delta = 0.1$ and we observe



(a) Approximation with degree $d = 50$

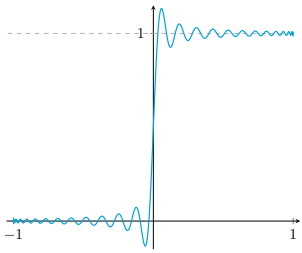


(b) Approximation with degree $d = 100$

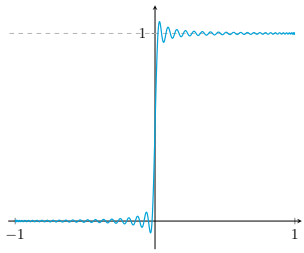


(c) Approximation with degree $d = 500$

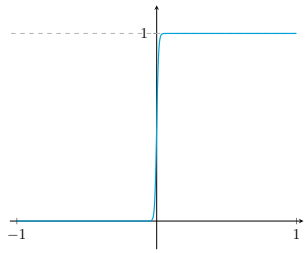
Figure 104: Chebyshev approximations of the raw $\text{sgn}(x)$ function in $[-1, 1]$, with different degrees



(a) Approximation with degree $d = 50$



(b) Approximation with degree $d = 100$



(c) Approximation with degree $d = 500$

Figure 105: Chebyshev approximations of the 150-scaled Sigmoid function in $[-1, 1]$, with different degrees

that, with $k = 120$, the error $\ell_\infty(\text{sgn}(x) - \text{sigmoid}_k(x))$ is negligible (i.e., less than 10^{-6}) in the interval $[-1, -\delta] \cup [\delta, 1]$. Using this function allows for much smoother approximations, since near $x = 0$ we can control the "steepness" of the jump. We report some illustrations of such approximations in Figure 105. The behavior of the polynomial for $x \in [-\delta, \delta]$ is irrelevant, since by assumption there will be no pair of elements such that its difference lies in such interval. Nonetheless, we still require a precise approximation for $0 \in [-\delta, \delta]$, which needs to be mapped to 0.5, although for that we have the following result.

Lemma 30

For any Chebyshev polynomial approximation $p(x)$ of the k-scaled Sigmoid function of degree $d > 0$ in the interval $[-1, 1]$, it holds that $p(0) = 0.5$, regardless of the scaling factor $k > 1$.

Proof. When approximating the scaled sigmoid function using Chebyshev polynomials, the behavior of the approximation depends on the parity of the polynomial degree d. Specifically, two distinct cases arise:

- Even degree d: remark that the $d+1$ Chebyshev interpolants x_i, from x_0 to x_d, are distributed as

$$x_i = \cos\left(\frac{(i+\frac{1}{2})\pi}{d+1}\right) \text{ with } 0 \leq i \leq d$$

Given the approximation interval $[-1,1]$, we will have $d+1$ interpolants, and the one "in the middle" is set at x_m, with $m = \frac{d}{2} \in \mathbb{Z}$.

$$\begin{aligned} x_m &= \cos\left(\frac{(m+\frac{1}{2})\pi}{d+1}\right) = \cos\left(\frac{(\frac{d}{2}+\frac{1}{2})\pi}{d+1}\right) \\ &= \cos\left(\frac{(\frac{d+1}{2})\pi}{d+1}\right) = \cos\left(\frac{\pi}{2}\right) = 0. \end{aligned}$$

Therefore, the polynomial $p(x)$ passing through the Chebyshev interpolants x_i will necessarily be equal to 0.5 in the origin since, by construction, $p(x_m) = \text{sigmoid}_k(x_m) = \text{sigmoid}_k(0) = 0.5$. Notice that $\text{sigmoid}_k(0) = 0.5$ for any $k > 1$.

- Odd degree d: Chebyshev interpolants are distributed equally between the first $[0,\ldots,(d-1)/2] \cap \mathbb{Z}$ and the second $((d-1)/2,\ldots,d]) \cap \mathbb{Z}$ set of roots.
Notice that the second set of interpolants can be written with respect to the first one by reversing signs and order, since for any $0 < i \leq d$:

$$\cos\left(\frac{(d-i+\frac{1}{2})\pi}{d+1}\right) = \cos\left(\pi + \frac{(i+\frac{1}{2})\pi}{d+1}\right) = -\cos\left(\frac{(i+\frac{1}{2})\pi}{d+1}\right)$$

which holds because $\cos(\pi+\theta) = -\cos(\theta)$. This implies that the interpolants are symmetric with respect to the y-axis since $-x_i = x_{d-i}$.
The polynomial passing through these interpolants will also pass through the middle point $(m, \text{sigmoid}_k(m))$ between the last point $p(x_{(d-1)/2})$ of the left interval and the first point $p(x_{1+(d-1)/2})$ of the second interval to keep the symmetry with respect to the y-axis, where m is:

$$\begin{aligned} m &= \left(\cos\left(\frac{((d-1)/2+\frac{1}{2})\pi}{d+1}\right) + \cos\left(\frac{(1+(d-1)/2+\frac{1}{2})\pi}{d+1}\right)\right)\cdot\frac{1}{2} = \\ &= \left(\cos\left(\frac{(d/2)\pi}{d+1}\right) + \cos\left(\frac{\pi+(d/2)\pi}{d+1}\right)\right)\cdot\frac{1}{2} \end{aligned}$$

Now, notice that the two angles sum to π:

$$\frac{(d/2)\pi}{d+1} + \frac{\pi + (d/2)\pi}{d+1} = \frac{\pi + d\pi}{d+1} = \frac{\pi(d+1)}{d+1} = \pi$$

as before we use that $\cos(\theta) + \cos(\theta + \pi) = 0$, meaning that the middle point is indeed $m = 0$, therefore $p(m) = \text{sigmoid}_k(m) = 0.5$. □

We can thus safely use Chebyshev approximations of the scaled sigmoid function, by carefully choosing a parameter $k > 1$ that allows for the scaled sigmoid to behave as $\text{sgn}(x)$ in the interval $[-1, -\delta] \cup [0] \cup [\delta, 1]$, with negligible errors.

Remark 10 (*Ascending and descending sorting*)

Notice that this framework is flexible enough to easily switch between ascending and descending indexing, which then will be reflected in the actual sorting, at no cost. At the beginning of the circuit, instead of computing $\boldsymbol{c}_{repeated} - \boldsymbol{c}_{expanded}$, simply compute the opposite $\boldsymbol{c}_{expanded} - \boldsymbol{c}_{repeated}$.

Building the corresponding permutation matrix. Let us for a moment ignore the eventual noise contained in the indexing vector $\boldsymbol{s} \in \mathbb{R}^n$ coming from approximate computations. Suppose having $\boldsymbol{v} = (4, 7, 1, 8)$ and its corresponding indexing $\boldsymbol{s} = (2, 3, 1, 4)$, as computed in the above presented procedure. The procedure to obtain a permutation matrix simply performs an equality check between the indexing vector and n vectors $\boldsymbol{i} \in \{i\}^n$, for $0 < i \leq n$. Again, with SIMD this can be efficiently parallelized. In [MEHP25], this equality check is performed by using the $\text{equal}(x)$ function – which they call $\text{Ind}_0(x)$ – that is eventually evaluated using approximations of the $\text{sgn}(x)$ function. In particular they observed that $\text{equal}(x)$ can be implemented as:

$$\text{equal}(x) := \text{sgn}\left(\frac{x + 0.5}{r}\right) \cdot \left(1 - \text{sgn}\left(\frac{x - 0.5}{r}\right)\right), \quad (7.5)$$

where the parameter $r > 0$ controls the range of values, and the 0.5 shift gives tolerance to accept values not necessarily very close to 0. We report a plot of this function in Figure 106. However, we observe that such solution is strongly nonlinear and may be excessively complicated, especially considering that it requires the computation of two $\text{sgn}(x)$ functions. Instead, we propose to relax this phase by considering the following observation.

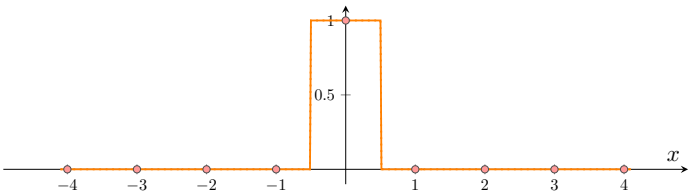


Figure 106: The approximation for equal(x) used in [MEHP25], for $a = 4$

Lemma 31

Any function $f : \mathbb{R} \to \mathbb{R}$ such that $f(0) = 1$ and $f(x) = 0$ for all $x \in \mathbb{Z} \setminus \{0\}$ can be used to construct a permutation matrix P from the indexing of a vector $\boldsymbol{v} \in \mathbb{Z}$ such that $P \cdot \boldsymbol{v}$ is sorted.

Notice that the definition of equal(x) in [MEHP25] respects the requirements of Lemma 31. Nevertheless, their approach is more generic as the approximation holds for all the values $x + \varepsilon \in \mathbb{R}$, with $x \in \mathbb{Z}$ and $\varepsilon \in (-0.5, 0.5)$ being an "error" term, which introduces some tolerance but at the same time might be excessively generic for some applications. Therefore, we propose to use a simple function that respect the barely minimum requirements of Lemma 31: the sinc(x) function, defined as:

$$\text{sinc}(x) := \frac{\sin(x\pi)}{(x\pi)} = \prod_{n=1}^{\infty} \left(1 - \frac{x^2}{n^2}\right). \tag{7.6}$$

The latter can be efficiently approximated, moreover (i) sinc$(0) = 1$ and (ii) sinc$(x) = 0$ for every $x \in \mathbb{Z} \setminus \{0\}$. We conveniently report its plot in Figure 107.

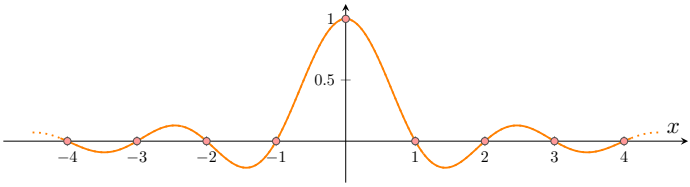


Figure 107: The sinc(x) function

Its error tolerance is much smaller than the one in (7.5), in particular sinc(x) approximate equal(x) for all the values $x + \varepsilon \in \mathbb{R}$ for very tiny values of $\varepsilon > 0$; for this reason we require the indexing vector to be as close as possible to $\mathbb{Z}$. We propose to use Chebyshev polynomials to approximate sinc(x) on the differences between the indexing $\boldsymbol{s}$ and each $\boldsymbol{i}$. This will lead to the permutation matrix P, encoded column-wise (Figure 108). The main drawback of choosing sinc(x) is that the values contained in the indexing phase must be very close to $\mathbb{Z}$. Nevertheless, The main advantage is that sinc(x)

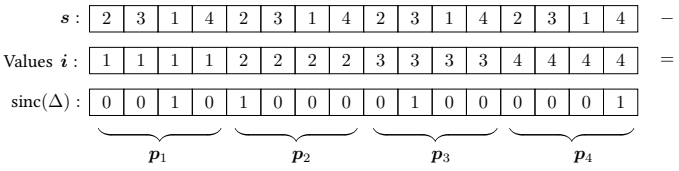


Figure 108: Generating the columns of P using the $\mathrm{sinc}(x)$ function over the differences between each pair $\boldsymbol{s}$ and $\boldsymbol{i}$.

can be well approximated without consuming many levels, making it suitable for low depth circuits. The indexing phase should be as precise as possible, then $\mathrm{sinc}(x)$ can handle the resulting values without requiring approximations of gigantic degrees.

Faster tie-correcting offset. The tie-correction offset in [MEHP25] is computed using some partial results obtained in the indexing phase (using the output of what they call the Eq function), while we propose to use $\mathrm{sinc}(x)$ at the beginning of the circuit, on the difference between the expanded and the repeated inputs. The advantages are that: (i) correcting the indexing for repeated elements comes at no cost in terms of circuit depth, since this phase is "parallel" to the indexing one, and (ii) we use the input values which contain the least amount possible of noise, and thus are supposed to be close to $\mathbb{Z}^4$ by assumption. On the other hand, the main drawback is that we compute the equality check based on $\mathrm{sinc}(x)$ from scratch, requiring some extra time – although as experiments will show, this can be computed very quickly and will have a limited impact in the final runtime.

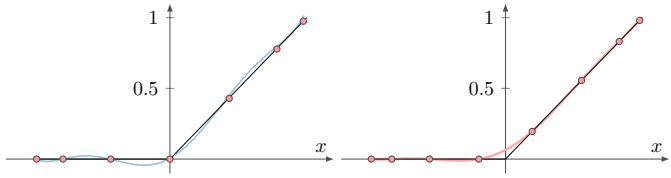
4: After a proper scaling by $1/\delta$, where δ is the maximum nonzero distance between two elements

7.3.2 Network-based sorting

The sorting network relies on comparisons between pairs of values. We propose to evaluate the comparison operation using the following relation:

$$\min(a, b) = a - \mathrm{ReLU}(a - b), \tag{7.7}$$

where $\mathrm{ReLU}(x)$ is approximated using Chebyshev polynomials. In Figure 109, we present the two different behaviors assumed by the polynomials: when the degree d is even, and when is odd. In general, increasing the degree of the polynomial does not always imply a smaller error.



(a) Even degree $d = 6$ **(b)** Odd degree $d = 7$

Figure 109: Approximations of the ReLU function, using Chebyshev interpolants

By computing the maximum error in $x \in [-1, 1]$ as $\ell_\infty(\text{ReLU}(x) - p_d(x))$, where $p_d(x)$ is the Chebyshev approximation with degree d, a much more stable approximation is obtained when d is odd, although the value $x = 0$ will have the largest error. This means that the network will slightly perform better if there are no repeated values – although this will not cause problems, at it will be shown in the experiments.

Table 13 presents the depth consumed by evaluating (7.2) using Chebyshev polynomials with respect to [Lee+22c] – remark that their approach is to evaluate the ReLU function using (7.1).

Precision bits	Consumed depth	
	[Lee+22c]	(**Proposed**)
6	6	5 (using $d = 20$)
7	7	6 (using $d = 38$)
8	8	7 (using $d = 76$)
9	9	8 (using $d = 156$)
10	11	9 (using $d = 310$)

Table 13: Approximations for the ReLU function. We compare to [Lee+22c, Table IV]

7.3.2.1 Evaluating a sorting layer via SIMD computations.

As it will be now shown, each layer of the sorting network can be computed in a single SIMD comparison. Therefore, we define the complexity of a sorting network as the number of its layers. In particular, given n inputs (with n being necessarily a power of two), the number of layers of the corresponding sorting network is equal to $(\log(n)(\log(n) + 1))/2$. As a consequence, the depth of our circuit scales at most like the logarithm squared of the number of input values n. The challenge is to build a circuit that evaluates a sorting network by performing the least possible amount of comparisons and by minimizing the number of consumed levels (i.e., multiplicative depth). We introduce an algorithm to evaluate a

sorting network layer (namely the *compare-and-swap* operation) using as example a sorting network for eight values. Without loss of generality, the algorithm can be generalized to evaluate any sorting network layer. Given the 1-Bitonic block in Figure 110, composed of a single layer, and a sample vector $\boldsymbol{c} := (5, 4, 1, 8, 2, 9, 1, 2)$. we evaluate it in four steps.

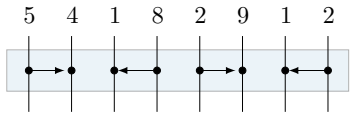


Figure 110: 1-Bitonic block with 8 input values

1. Obtain the minimum between each pair (refer to pairs of elements connected by the arrows in Figure 101) as shown in (7.7):

$$\boldsymbol{c}_1 = \underbrace{\boldsymbol{c}}_{a} - \underbrace{p\big(\boldsymbol{c} - \mathrm{Rot}_i(\boldsymbol{c})\big)}_{\mathrm{ReLU}(a-b)}$$

   where $p(x)$ is some polynomial approximation of the ReLU(x) function (in our experiments we use Chebyshev polynomials, but any approximation can be plugged in), $\boldsymbol{c}$ is the encrypted input vector to be sorted and i is the distance between compared elements; in this example, this is equal to 1. Referring to Figure 99, this operation gives as output all wires containing $\min(a, b)$.

2. Obtain the maximum between each pair: recalling that $\max(a, b) = a + b - \min(a, b)$, it is possible to obtain these values as:

$$\boldsymbol{c}_2 = \big(\underbrace{\boldsymbol{c}}_{a} + \underbrace{\mathrm{Rot}_{-i}(\boldsymbol{c})}_{b} \big) - \underbrace{\mathrm{Rot}_{-i}(\boldsymbol{c}_1)}_{\min(a,b)}$$

   This gives us the comparisons between the values that in Figure 101 are compared by an arrow pointing down. Considering the previous example, we computed the values in the bottom part of following the 1-bitonic block (Figure 111).

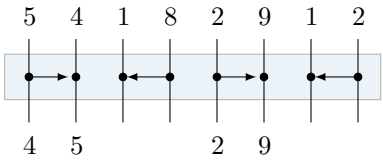


Figure 111: Values obtained in $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$

   In particular, $\boldsymbol{c}_1$ contains the minimum values (4 and 2), while $\boldsymbol{c}_2$ the maximum ones (5 and 9).

3. Is is not necessary to evaluate the minimum function again, since the minimum values of the remaining wires (whose arrows are pointing left) have already been computed and are available in $\boldsymbol{c}_1$; they just need to be rotated:

$$\boldsymbol{c}_3 = \text{Rot}_{-i}(\boldsymbol{c}_1)$$

4. As before, the maximum values of the remaining wires are computed using the previously computed minimum:

$$\boldsymbol{c}_4 = (\boldsymbol{c} + \text{Rot}_i(\boldsymbol{c})) - \boldsymbol{c}_1$$

At this point, all the required elements have been computed, and can be obtained by summing up the four ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ and $\boldsymbol{c}_4$ (Figure 112).

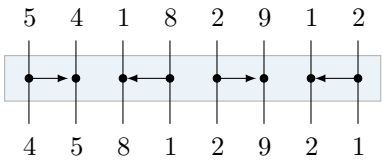


Figure 112: Complete evaluation of a sorting layer

However, before doing that, unnecessary values, for each partial ciphertext $\boldsymbol{c}_i$, must be cancelled (i.e., multiplied by zero), otherwise incorrect values will ruin the results. Think, as an example, to the second slot of $\boldsymbol{c}_1$, which will contain partial computations to be removed before the sum. The EvaluateLayer procedure is formally presented in Algorithm 12.

```
1 function EvaluateLayer (c, n, α, b, ℓ)
2     c_1 := min(c, Rot_α(c))
3     c_2 := (c + Rot_{-α}(c)) − c_1
4     c_3 := Rot_{-α}(c_1)
5     c_4 := (c + Rot_α(c)) − c_1
6     m_1, m_2, m_3, m_4 := GenMasks(n, b, ℓ)
7     return Add(Mul(c_i, m_i)) for all 0 < i ≤ 4
```

Algorithm 12: Evaluation of a sorting network layer

This procedure takes as input a ciphertext $\boldsymbol{c}$, the value of α (namely the length of the arrows), the current bitonic block b and the layer $0 < \ell \leq b$ of the current block. It computes the four ciphertexts $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ and $\boldsymbol{c}_4$, implementing the procedure described above. It relies on the GenMasks procedure, described in Algorithm 13, which returns a set of four masks to be applied to the different $\boldsymbol{c}_i$.

```
1  m_1, m_2, m_3, m_4 := empty vectors
2  function GenMasks (n, b, ℓ);
3      while size(m_1) < n do
4          for 2^ℓ times do
5              for 2^{b−ℓ} times do
6                  PushMask(1);
7              for 2^{b−ℓ} times do
8                  PushMask(2);
9          if size(m_1) + 2^b ≥ n then
10             break;
11         for 2^ℓ times do
12             for 2^{b−ℓ} times do
13                 PushMask(3);
14             for 2^{b−ℓ} times do
15                 PushMask(4);
16     return m_1, m_2, m_3, m_4;
17 function PushMask (idx);
18     for j := 1 to 4 do
19         if j = idx then
20             push 1 to m_j;
21         else
22             push 0 to m_j;
```

Algorithm 13: Generation of the masks for the comparisons c_i

This set of masks strongly depends on the current network layer, and they are constructed according to the length of the arrows and their direction. Putting these procedures together, it is possible to build an SIMD-based encrypted bitonic sorting circuit (Algorithm 14).

```
1 function NetworkSort (c);
2     n := length of c;
3     for b := 1 to ⌈log(n)⌉ do
4         for ℓ := 1 to b + 1 do
5             α := 2^{(i−j)};
6             c := EvaluateLayer(c, n, α, b, ℓ);
```

Algorithm 14: SIMD-based bitonic sorting of $\boldsymbol{c}$

In our setup, each EvaluateLayer operation is followed by a CKKS bootstrapping operation on $\boldsymbol{c}$ which refreshes the level

of the ciphertext and enables the evaluation of the subsequent layer.

7.4 Experiments

As above, we split this section in two parts: in the first we present experiments based on the permutation-based approach, in the second one on the network-based approach. Each experiment uses the OpenFHE implementation of the RNS-CKKS [Al+22; KPP22] scheme, a security level of $\lambda > 128$ bits, rings of size $N = 2^{16}$ and sparse ternary secret key distribution. All experiments have been run on a Macbook 2021 M1 Pro laptop with 16 GB of memory and are replicable by running the corresponding script in the experiments folder, contained in our open-source repository[5]. The latter also contains Python notebooks that implement the circuits executed on clear data, useful to better understand their workings. The parameters have been chosen in such a way that sorting of uniform random values, sampled from $[0, 1]$ at minimum distance δ is correct. Formally, given the clear vector $\boldsymbol{v} \in \mathbb{R}^n$ and the decrypted vector (after the evaluation of the circuit) $\boldsymbol{c} \in \mathbb{R}^n$, we say that c_i is correct if its absolute difference with v_i is smaller than δ, i.e., if

$$|c_i - \text{sort}(\boldsymbol{v})_i| < \delta.$$

The sorting is correct if this holds for all c_i, with $0 < i \leq n$.

[Al+22]: Al Badawi, Bates, Bergamaschi, Cousins, et al., *'OpenFHE: Open-Source Fully Homomorphic Encryption Library'* . In: WAHC '22

[KPP22]: A. Kim, Papadimitriou, and Polyakov, *'Approximate Homomorphic Encryption with Reduced Approximation Error'* . In: CT-RSA '22

5: github.com/lorenzorovida/Lightweight-Sorting-In-Approximate-Homomorphic-Encryption

7.4.1 Permutation-based

The main tunable parameters used in these experiments are: (i) the degree d_1 of the approximation of the sigmoid$_k(x)$ function, (ii) its scaling factor k, and (iii) the degree of the approximation of the sinc(x) function d_2.

Concerning the tie correction offset, we use an approximation of the sinc(x) function of degree of $d_{\text{tie}} < d_1$ so that this phase does not increase the circuit depth, as it can be performed in parallel with the indexing phase[6]. The evaluation of the tie-correction offset does not have a huge impact in the final computation in terms of runtime – just a few seconds – but by being very precise it allows to correct the offset without introducing a significant error in the indexing vector, which, we stress, should be as close as possible to $\mathbb{Z}$. In order to

6: Taking into account a product by 0.5, so the degree of sinc(x) should allow for one extra multiplication.

Table 14: Parameters used in the first permutation-based sorting. Degrees d_1, d_2, d_{tie} refer to the Chebyshev polynomials approximations

	CKKS parameters			Data parameters			
n	Δ	Circuit depth	$\log(QP)$	$\text{sigmoid}_k(x)$ scale k	$\text{sigmoid}_k(x)$ degree d_1	$\text{sinc}(x)$ degree d_2	$\text{sinc}(x)$ degree d_{tie}
8	2^{45}	17	1233	650	1006	59	495
16	2^{45}	17	1233	650	1006	59	495
32	2^{45}	18	1338	650	1006	119	495
64	2^{45}	19	1383	650	1006	247	495
128	2^{45}	20	1488	650	1006	495	495

consume less levels, all Chebyshev polynomial approximations are evaluated in $[-1, 1]$, with inputs and outputs scaled accordingly. The CKKS parameter which controls the complexity of decomposition during the key switching, is set to the minimum possible $d_{\text{num}} = 2$ (refer to [KPZ21] for a detailed explanation of the d_{num} parameter in key switching). Notice that higher values of d_{num} allow to evaluate deeper circuits without reducing the security level, at the cost of increasing computational complexity (larger d_{num} implies smaller modulus $\log(QP)$, which with the ring size N define the security of the underlying RLWE problem), although we do not require to reduce the size of the modulus, as our circuits are already shallow.

Remark 11

The polynomial approximations in the actual implementation are always performed without loss of generality over the $[-1, 1]$ interval, as this allows to save one multiplication. The indexing vector will thus not live in $\mathbb{Z}^n$ but in $\delta\mathbb{Z}^n$. This is simply an implementation choice that does not change the output, but makes the computation faster.

Experiment 1, comparison with [MEHP25] on single ciphertexts. In the first experiment, we define the same setup as in [MEHP25] in single-ciphertext mode, namely number of values n ranging from 8 up to 128 – our small ring does not allow to encrypt up to 256 values – and minimum distance between pair of elements $\delta = 0.01$, with tie-correction active. The degree d_2 of the $\text{sinc}(x)$ approximation depends on the number of values n (remark that this function is evaluated on the indexing vector), while the degrees d_1 and d_{tie}, along with the scaling factor k, depend on δ. For every run, values

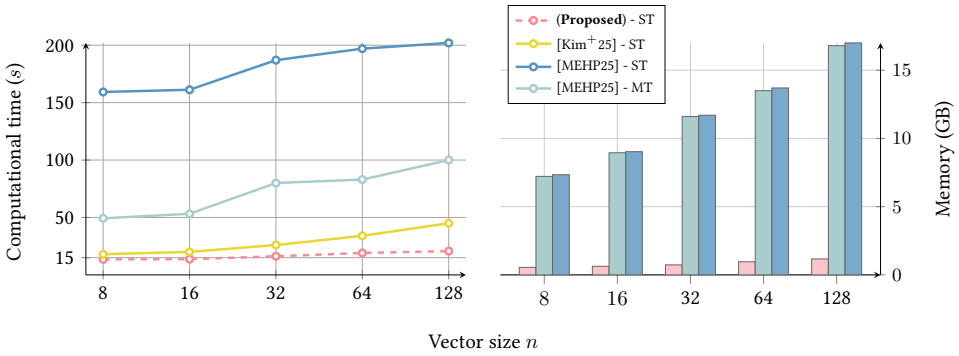


Figure 113: Comparison on computational time and memory consumption between our proposal and [MEHP25; Kim+25]. ST and MT stand for single and multithread. [Kim+25] did not report memory requirements

have been sampled randomly from a discretization of $[0, 1]$, with step size equal to δ, with possible repeated values, whose indexing is corrected by the tie-correction offset. We present in Table 14 the parameters used in the various tests.

As reported in Figure 113, computational times and memory requirements are always lower than in [MEHP25; Kim+25], despite our modest hardware (in [Kim+25] they employ an Intel Xeon Gold 6226R). The main factor that reduces both time and memory is the choice of the ring $N = 2^{16}$, which can be safely used since the levels required by our circuit are at most 20. All the chosen moduli QP are still notably below the 128-bits of classical security threshold – roughly 2^{1772} – established by the Lattice estimator [APS15].

Curiously, our runtimes and memory requirements does not grow much with respect to the number of elements n. This is a natural consequence of the fact that the complexity of the circuit mainly depends on the factor δ, which in turn defines the degree d_1 of the polynomial approximations of $\mathrm{sgn}(x)$, which is the main bottleneck of the circuit. On the other hand, increasing n results in larger degrees d_2, which are always small with respect to the ones used for $\mathrm{sigmoid}_k(x)$.

Experiment 2, more discretization. We now aim to increase the granularity of the considered $[0, 1]$ interval from $\delta = 0.01$ to $\delta = 0.001$. In order to achieve a correct sorting, we must increase the degrees of approximations of the nonlinear functions, in particular of $\mathrm{sigmoid}_k(x)$. Luckily, from the previous experiment, we still have room to increase the circuit depth maintaining $\lambda > 128$ bits of security without requiring

[Kim+25]: S. Kim, Ünay, Yilmazer-Metin, and H. T. Lee (2025), Optimized Rank Sort for Encrypted Real Numbers (Cryptology ePrint Archive, Paper 2025/1170)

[APS15]: M. R. Albrecht, Player, and Scott (2015), *'On the concrete hardness of Learning with Errors'* . In: Journal of Mathematical Cryptology

Table 15: Parameters used in permutation-based sorting experiments. Degrees d_1, d_2 refer to the Chebyshev polynomials approximation

	CKKS parameters			Data parameters			
n	Δ	Circuit depth	$\log(QP)$	$\text{sigmoid}_k(x)$ scale k	$\text{sigmoid}_k(x)$ degree d_1	$\text{sinc}(x)$ degree d_2	$\text{sinc}(x)$ degree d_{tie}
8	2^{45}	21	1533	9170	16000	59	495
16	2^{45}	21	1533	9170	16000	59	495
32	2^{45}	22	1638	9170	16000	119	495
64	2^{45}	23	1683	9170	16000	247	495
128	2^{45}	24	1728	9170	16000	495	495

an increment in the ring size. We thus increase the degree d_1 of the scaled sigmoid function to $d_1 = 16000$, with the scaling factor set to $k = 9170$; refer to Table 15 and Figure 114. Results

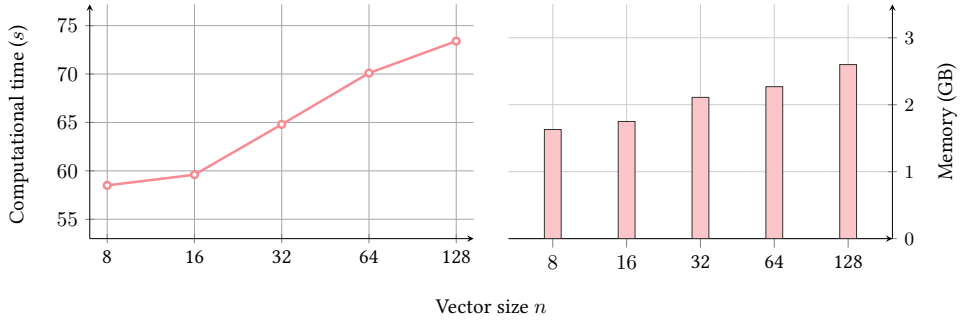


Figure 114: Time and memory requirements when sorting elements at minimum distance $\delta = 0.001$

confirm that the runtime is somehow stable regardless of the vector size n; it only fluctuates by a few seconds at most. Also memory requirements are always around 2 GB, never larger than 3 GB.

7.4.2 Network-based

We now aim to perform similar experiments as above, but based on the bitonic sorting network. Notice that the key material of the scheme required in such experiments will be drastically larger – and in turn, larger memory requirements – as a consequence of the bootstrapping operation, run after each layer evaluation. This also requires larger values of Δ that allow to reduce the new errors introduced by the bootstrapping procedure

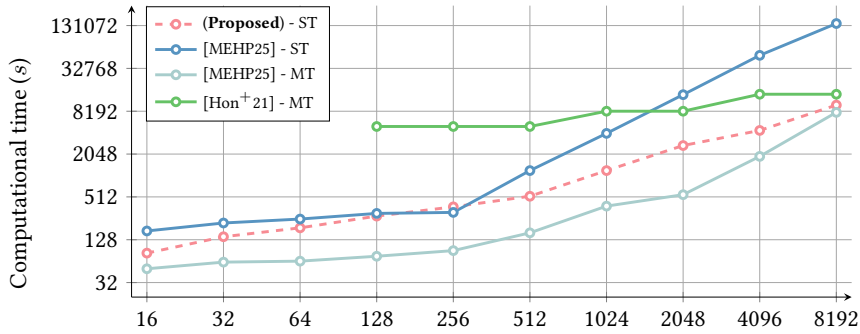


Figure 115: Comparing our network-based approach with [MEHP25; Hon+21] in terms of runtime

Experiment 1, sorting at distance $\delta = 0.01$. Differently from before, we can now sort up to 8192 elements without relying on multiple ciphertexts, since the number of required slots by the sorting network is simply $\Theta(n)$. The set of parameters is fixed for any value of n, since the only nonlinear function is ReLU(x), whose precision depends on δ. We present in Table 16 the parameters used in this phase.

	CKKS parameters				Data parameters
n	Δ	Circuit depth	$\log(QP)$	Bootstrap depth	ReLU(x) degree d
From 8 to 8192	2^{55}	25	1765	16	495

Table 16: Parameters used in network-based sorting experiments.

We present in Figure 115 the results comparing to the permutation-based sorting approach and to [MEHP25]. We also include, for the sake of completeness, a comparison with the CKKS-based sorting network in [Hon+21], based on approximations of the sgn(x) function.

Our network-based approach does not represent a significant improvement with respect to the permutation-based sorting in [MEHP25], although it is orders of magnitude faster than the network in [Hon+21]. We observe that during the experiments we have exceeded the laptop available memory, leading to virtual memory usage, which eventually have led to overestimated runtimes, although we are at this point interested to rough estimates. Despite slower, it can be considered as a decent alternative to permutation-based as it can be used on large amounts of elements without assuming to have n^2 slots available.

Even though the network-based approach requires larger amounts of memory with respect to the permutation-based

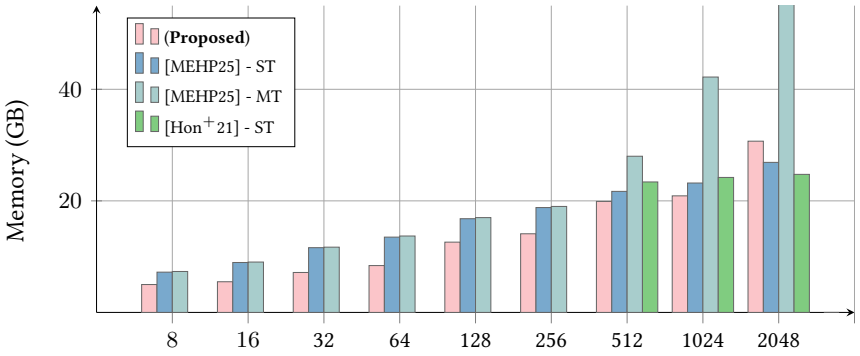


Figure 116: Comparing our network-based approach with [MEHP25; Hon+21] in terms of memory. The last cut bar is valued 97 GB, cut for readability

one (mainly because of the bootstrapping operation), the $N = 2^{16}$ ring allows to stay competitive with [MEHP25], as illustrated in Figure 116.

7.5 Conclusion

We proposed two approaches to sort encrypted values under the CKKS scheme. In particular, the permutation-based approach is very quick and do not have large memory requirements, but it has the drawback of requiring $\Theta(n^2)$ slots in order to efficiently evaluate the SIMD operations. On the other hand, the network-based approach is slower and requires more memory, but has the advantage of requiring only $\Theta(n)$ slots, making it more suitable in certain applications. It would be interesting to explore different ways to tackle the procedures in the permutation-based sorting – using $\text{sinc}(x)$ is just one of them. The ultimate goal would be to be able to run a permutation-based sorting without resorting to $\text{sgn}(x)$. It would be also interesting to explore new functions to use in the sorting network, different from $\text{ReLU}(x)$, which might lead to shorter circuits and lighter computations.

An interesting open point is about dealing with large amount of values in the permutation-based approach. In [MEHP25], they propose to split the input vector in multiple ciphertexts, while one idea for instance could be to build a hybrid sorting algorithm where each subvector is partially sorted using permutations, and the whole vector is reconstructed by evaluating only the last block of a bitonic network, which takes as input sequences that are partially sorted.

Final conclusions, remarks and questions

8

The goal of this thesis has been to present all the findings I conducted, in collaboration with my co-authors, over the course of my three-years Ph.D. In particular, we tackled FHE from two different perspectives: constructions and algorithms. The main contributions are presented in Chapter 4 for constructions, and Chapter 5, 6 and 7 for algorithms. I now provide a general perspective – from a personal point of view – to place these contributions within the broader context of Computer science.

8.1 Why?

We live in a world where *everything* is connected, tracked, and logged every day. Every (personal) information we enter in any website is constantly at risk, hence we highly believe that this is an issue that requires an urgent, strong, technical and provable solution; the latter should not be delegated to laws only (e.g., [EC16b]). Since the advent of online services based on artificial intelligence, this risk is even more critical. As an example, today it is possible to use a web browser fully controlled by an AI, that helps and enhances our navigation. Nevertheless, as it stands, everything we do through a browser is sent and available to an external service provider, that we should trust. As cryptographers, we believe that trusting an external (and usually opaque) service provider is not enough, and actual cryptographic solutions should always be brought to real world scenarios, in a transparent and open-source way (e.g., Signal[1]). This allows us to "shift" the trust from any – usually for-profit – entity that controls our data, to a community of security and privacy experts. We suggest an interesting paper that discusses this shift of trust [BNP22].

[EC16b]: European Parliament and Council of the European Union (2016), Regulation (EU) 2016/679 of the European Parliament and of the Council (online)

1: https://github.com/signalapp

[BNP22]: Balsa, Nissenbaum, and S. Park, *'Cryptography, Trust and Privacy: It's Complicated'* . In: CSLAW '22

Additionally, cryptographic solutions typically provide answers to many other urgent questions, think of, as an example, the recent proposal by the EU commission [Eur22] to contrast the child sexual abuse online. As many other cryptographers stated in an open letter[2], we can not think of solving these issues by removing privacy from people's lives – especially when cryptography gives us tools that allows one to tackle these problems in a privacy-preserving way. This is why cryptography is necessary and improvements in this field directly correspond to improvements in people's privacy, which is a

[Eur22]: European Commission (2022), Proposal for a Regulation of the European Parliament and of the Council laying down rules to prevent and combat child sexual abuse (COM(2022) 209 final / 2022/0155(COD)) (online)

2: https://csa-scientist-open-letter.org/Sep2025

widely recognized fundamental right. As such, it is important to never forget that these rights do not come for granted, but must be actively protected, strengthened, and defended through continuous effort, innovation, and vigilance.

8.2 Future work and open questions

Construction of FHE schemes. In Chapter 4 we were able to present a framework that allows one to build a FHE scheme starting from any hardness assumption that hides the short vector of a lattice, although many questions remain open. Is LWE the best choice, after all? Our findings showed that LWE is indeed very efficient and optimized, and we demonstrated it from a geometrical perspective. Nonetheless, this approach has been studied since twenty years, while, e.g., LIP is a pretty new assumption and still needs time. Additionally, the lattice perspective offered by LIP might unlock different approaches that can solve open problems that might be artifacts of LWE. As an example, the key switching under LWE does not take fully advantage of lattice operators. Some future research direction might be about a different definition of key switching, from one lattice to another, as a projection. Also, we did not discuss bootstrapping, although this could be another key element that can be done differently under different assumptions. First of all, decoding using the short primal vector is a "standard" procedure in lattice-based cryptography, but not the only one. Can different decryption circuits make bootstrapping faster?

Another interesting point is about making the schemes we presented (i.e., Lattice-BGV and Lattice-GSW) more efficient. Just like RLWE for LWE, it would be very interesting to investigate structured lattices to make the schemes faster and less complex from a computational point of view. As a reference, think of the signature scheme HAWK [Bos+23] that, using module-LIP, gave a fresh perspective on lattice-based signatures that outperformed the state-of-the-art proposals based on lattices. Could we have something like this, in the world of FHE?

[Bos+23]: J. W. Bos, Bronchain, Ducas, Fehr, et al. (2023), HAWK . In: Submission to the NIST's post-quantum cryptography standardization process

Algorithms based on FHE schemes. In the subsequent Chapters 5, 6 and 7 we presented some solutions for algorithms based on the CKKS scheme. These works share the same main motivation: FHE algorithms are very inefficient and need optimizations to close the gap to real-world applications.

Much of the energy has been dedicated to privacy-preserving applications, which we hope one day could be the norm. Chapter 5 presented a solution to the problem of classifying encrypted images. As shown in the results, although competitive and under some aspects better than the state-of-the-art, performance is still far from being deployable in real world applications. Some future research directions should include a dialogue between the machine learning and the FHE communities. For example, building networks specifically designed taking into account how FHE works (e.g., following the example of [KMS25]), could represent a strong improvement over how we build encrypted networks.

[KMS25]: Kern, Manino, and Sinz, *'Certified Error Analysis of Homomorphically Encrypted Neural Networks'* . In: SAIV '25

In Chapter 6 we discussed about a possible implementation of Transformer architecture under FHE. The rise of LLMs made this effort required, as after the release of many generative AI-based chatbots, these entered our daily lives. Among the endless list of ethical issues and questions posed by these tools [SE24], we also have the privacy one. OpenAI recently released a blogpost[3] where they claim that "*[. . .] around 0.15% of users active in a given week have conversations that include explicit indicators of potential suicidal planning or intent*". We recognize the importance of raising these issues, and we strongly believe that they must be discussed and addressed. However, we must ask ourselves whether any entity should have unrestricted access to this kind of extremely personal user data. Again, cryptography has many (still theoretical) tools that would allow to draw these conclusions, without actually being able to access people conversations, in clear. Of course it is hubristically optimistic to think that, for example, an AI-based chatbot could be run *entirely* over encrypted tokens – as for the technology available today. Nevertheless, it is what we should aim for.

[SE24]: Stahl and Eke (2024), *'The ethics of ChatGPT–Exploring the ethical issues of an emerging technology'* . In: International Journal of Information Management

3: https://openai.com/index/strengthening-chatgpt-responses-in-sensitive-conversations

Lastly, in Chapter 7, we discussed some optimizations for sorting encrypted numbers. We can define a number of applications in which sorting plays a fundamental role, for example in the context of Private information retrieval. Being able to send an encrypted query and obtain an encrypted result is, today, something that can be done with the tools that cryptography gives us. Nonetheless, most of these tools are still slow and computationally expensive, and are not mature enough to be deployed in real world scenarios. Nonetheless, we are just defining the first steps towards a (hopefully) fully encrypted Internet communications, where service providers, including search engines, are able to offer the same services to the users, without impacting their privacy.

Some of the presented works are still under development, with further efforts being made to hopefully establish them as standard benchmarks in the homomorphic encryption community[4]. It is also gratifying to see that some of the research outcomes are featured by OpenFHE as valuable resources[5].

4: https://github.com/fhe-benchmarking

5: https://github.com/openfheorg/awesome-openfhe

Acronyms

This list describes the several acronyms used throughout the thesis.

ABE	Attribute-based encryption
AES	Advanced encryption standard
AGCD	Approximate greatest common divisor
BDD	Bounded distance decoding (refer to Definition 16)
BERT	Bidirectional Encoder Representations from Transformers (refer to [Dev+19])
BFV	Brakerski/Fan-Vercauteren (authors, refer to the scheme presented in [Bra12; FV12])
BGV	Brakerski-Gentry-Vaikunthanathan (authors, refer to the scheme presented in [BGV12])
BKZ	Block-Korkine-Zolotarev
BN	Batch normalization
BV	Type of key switching, Brakerski-Vaikunthanathan (authors, refer to the scheme presented in [BV11])
CDH	Computational Diffie-Hellman
CGGI	Chillotti-Gama-Georgieva-Izabachène (authors, refer to TFHE)
CIFAR	Canadian Institute For Advanced Research
CKKS	Cheon-Kim-Kim-Song (authors, refer to Section 3.3)
CNN	Convolutional neural network
CRT	Chinese remainder theorem
CVP	Closest vector problem (refer to Definition 15)
DCRT	Double Chinese remainder theorem representation
DFT	Discrete Fourier transform
DGHV	van Dijk-Gentry-Halevi-Vaikuntanathan (authors, refer to the scheme presented in [vDGHV10])

DH	Diffie-Hellman
DLOG	Discrete logarithm
DM	Ducas-Micciancio (authors, refer to FHEW)
FE	Functional encryption
FFT	Fast Fourier transform
FHE	Fully homomorphic encryption
FHEW	Fastest Homomorphic Encryption in the West (refer to the scheme presented in [DM15])
GapSVP	Gap-Shortest vector problem (refer to Definition 13)
GDPR	General Data Protection Regulation [EC16a]
GELU	Gaussian Error Linear Unit
GHS	Type of key switching, Gentry-Halevi-Smart (authors, refer to the scheme presented in [GHS12b])
GLWE	Group Learning with errors
GPU	Graphics Processing Unit
GSO	Gram-Schmidt orthogonalization (refer to Definition 6)
HE	Homomorphic encryption
HEAAN	Homomorphic Encryption for the Arithmetic of Approximate Numbers, also referred to as CKKS
HKZ	Hermite-Korkine-Zolotarev
IBE	Identity-based encryption
IDFT	Inverse discrete Fourier transform
IND-CPA	Indistinguishability under adaptive chosen-plaintext attack
INTT	Inverse Number theoretic transform
ISIS	Short integer solution (refer to Section 2.2)
LHE	Leveled homomorphic encryption
LIP	Lattice isomorphism problem
LLL	Lenstra-Lenstra-Lovasz (refer to Section 2.1.2)

LLM	Large language model
LN	Layer normalization
LSB	Least significant bit
LWE	Learning with Errors
ML	Machine learning
MLaaS	Machine Learning as a Service
MNIST	Modified National Institute of Standards and Technology database
MPC	Multi-party computation
MSB	Most significant bit
NLP	Natural language processing
NP	The fundamental complexity class that contains all the decision problems that can be solved by a nondeterministic Turing machine using a polynomial amount of computational time and verified by a deterministic Turing machine
NTT	Number theoretic transform
OWF	One-way function
P	The fundamental complexity class that contains all the decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computational time
PIR	Private information retrieval
PKC	Public-key cryptography
PKE	Public-key encryption
PPML	Privacy-preserving machine learning
PPT	Probabilstic-polynomial-time
ReLU	Rectified Linear Unit
ResNet	Residual neural network (refer to [He+16b])
ResNet20	Residual neural network with 20 layers
RLWE	Ring learning with errors (refer to Section 2.4.3)
RNS	Residue number system

RSA	Rivest-Shamir-Adleman (refer to the scheme presented in [RSA78])
SDP	Syndrome decoding problem (refer to Definition 23)
SIMD	Single Instruction, Multiple Data
SIS	Short integer solution (refer to Definition 29)
SSS	Shamir secret sharing
SST-2	Stanford Sentiment Treebank
SVP	Shortest vector problem (refer to Definition 12)
SWHE	Somewhat homomorphic encryption
TEE	Trusted execution environment
TFHE	Fast Fully Homomorphic Encryption over the Torus (refer to the scheme presented in [CGGI20])
uSVP	Unique-Shortest vector problem (refer to Definition 14)

Author's publications

This thesis is based on the following papers.

[LRvW25] A. Leporati, L. Rovida, and W. van Woerden. 'Beyond LWE: a Lattice Framework for Homomorphic Encryption'. In: *IACR ePrint Archive 2025/1171* (2025) (cited on pages 10, 12).

[RL24a] L. Rovida and A. Leporati. 'Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption'. In: *International Journal of Neural Systems* 34.05 (2024), p. 2450025 (cited on pages 12, 13).

[RL24b] L. Rovida and A. Leporati. 'Transformer-based Language Models and Homomorphic Encryption: An Intersection with BERT-tiny'. In: *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics*. 2024, pp. 3–13 (cited on pages 12, 14).

[RLB25] L. Rovida, A. Leporati, and S. Basile. 'Lightweight Sorting in Approximate Homomorphic Encryption'. In: *IACR ePrint Archive 2025/1150* (2025) (cited on pages 12, 15).

During the Ph.D., the author has additionally co-authored the following papers, which are not discussed in the thesis.

[LR24] A. Leporati and L. Rovida. 'Looking for stability in proof-of-stake based consensus mechanisms'. In: *Blockchain: Research and Applications* 5.4 (2024), p. 100222.

[LR25] A. Leporati and L. Rovida. 'An evolutionary approach to the design of spiking neural P circuits'. In: *Journal of Membrane Computing – Special Issue Devoted to CMC 2024* (Feb. 2025), pp. 1–15.

[Rov23] L. Rovida. 'Fast but approximate homomorphic k-means based on masking technique'. In: *International Journal of Information Security* 22.6 (Dec. 2023), pp. 1605–1619.

Acknowledgements

Science is a team effort, and I strongly believe that. However, I also believe that "team" in a broader sense includes many people, not just scientists. I cannot think of anything I could have learned and achieved in this field without the many people who have supported me since the beginning, from school teachers to university professors. I want to thank my supervisor Alberto for introducing me to the world of cryptography and giving me the opportunity of pursuing this Ph.D., without never putting me under extra pressure. Also thanks to Luca and his advices for the beginning of my Ph.D. Additionally, I would like to thank all the co-authors of all the findings we were able to obtain.

Of course being a student means that someone keeps things running, and for that I would like to thank the whole department of Computer science at the University of Milan-Bicocca, and in particular to the always available coordinator Leonardo, and to all the professors that made this journey extremely enjoyable. Nonetheless, secretaries made it easy, so thanks Leo, Letizia and all the others. During these years I've been also learning through teaching. I would like to thank all my teaching colleagues, and in particular Daniela that trusted me and gave me the opportunity to teach in her course.

Pursuing a Ph.D. is not easy, but luckily with nice people around you things get way less difficult. I would like to thank all the people from *piano terra*, including (among others!) Alessia, Alidou, Caterina, Chiara, Davide, Fay, Federica, Giorgio (x2), Giulio, Gloria, Jacopo, Joe, Lucia, Marco, Michele, Renzo and Riccardo. Thank you for all the laughs, breaks, beers and spritz! I hope the best for you in the future.

Another nice aspect of being a Ph.D. student is the ability to travel, and in particular of having research periods abroad. I want therefore to warmly say merci beaucoup (or should I say dank u wel?) to Wessel for hosting me in the special Bordeaux, and for all the people from IMB welcoming me there, including Afonso, Audrey, Gabriel, Guilhem, Julien, Manolo and all the others! I would like to give a particular thank you to Wessel for all the time and energy he dedicated to me, teaching me so many stuff about lattice-based cryptography, sharing with me so much knowledge and experience that I will always carry with me.

Secondly, I would like to say duizendmaal dank to Frederik for hosting me in Leuven. The town and COSIC are very beautiful places, and I really enjoyed staying there. I look forward to be back someday! In particular, I would like to thank Gioella and Riccardo for first welcoming me, and then to all the people that I was able to meet when I was there – too many to be listed – including the Metafoor pub for friday beers. A particular thank you to Frederik, Jiayi and Robin for dedicating me time for discussions and for teaching me how to be a better researcher, and for guiding me in all the complexities of algebra in FHE! Also thanks to Daniele, Giulia and Samuele for the nice times (and hosting) in Bruxelles. Last but not least, thank you Péla for making everything super easy (and colorful!).

Hopefully, I will be able in the future to return all the knowledge, kindness and patience that I received in these years!

I would also like to extend the “team” to include my friends and family. Thanks to my parents for giving me the freedom of choosing what I liked to do, and for helping me in making it possible. Thanks to my brother, Mirko, and Eleonora for always being by my side and for all the dinners, laughs and happiness in these years. Thanks to Elena for sharing with me your creativity and energy for Versezero (looking forward to finish it!).

I want to say thanks to all the people I grew up with and to all the ones that accompanied me during (and not only) these last years: Alice, Andrea, Anita, Beatrice, Cate, Costanza, Davide, Emma, Fang, Gabriele, Gianluca, Giorgio, Irene, Karen, Luca, Ludovico, Mattia (x2), Nicolò, Roberta, Sam and Sara. A particular thanks to Cate and Memi for all the spilled tea, always with cautela.

Lastly, one of the main and constant sources of happiness and love in my life: thank you Franci for always being by my side and for never failing in making me feel motivated and loved in these years. With you it has been much easier.

Grazie a tutti!

October ’25 *Lollo*

Bibliography

The references in alphabetical order.

[AB25] W. Ao and V. N. Boddeti. 'CryptoFace: End-to-End Encrypted Face Recognition'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2025, pp. 19197–19206 (cited on page 11).

[ABD16] M. Albrecht, S. Bai, and L. Ducas. 'A Subfield Lattice Attack on Overstretched NTRU Assumptions'. In: *Annual International Cryptology Conference*. 2016, pp. 153–178 (cited on page 178).

[AD97] M. Ajtai and C. Dwork. 'A public-key cryptosystem with worst-case/average-case equivalence'. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. 1997, pp. 284–293 (cited on page 19).

[Ajt96] M. Ajtai. 'Generating hard instances of lattice problems (extended abstract)'. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. 1996, pp. 99–108 (cited on pages 10, 19, 59, 62, 64, 145).

[Ajt98] M. Ajtai. 'The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)'. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. 1998, pp. 10–19 (cited on pages 31, 33).

[Akl11] S. G. Akl. 'Bitonic Sort'. In: *Encyclopedia of Parallel Computing*. 2011, pp. 139–146 (cited on page 239).

[AKP25] A. Alexandru, A. Kim, and Y. Polyakov. 'General Functional Bootstrapping Using CKKS'. In: *Advances in Cryptology – CRYPTO 2025*. 2025, pp. 304–337 (cited on page 131).

[Al +21] A. Al Badawi, C. Jin, J. Lin, C. F. Mun, S. J. Jie, B. H. M. Tan, X. Nan, K. M. M. Aung, and V. R. Chandrasekhar. 'Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data With GPUs'. In: *IEEE Transactions on Emerging Topics in Computing* 9.3 (2021), pp. 1330–1343 (cited on page 185).

[Al +22] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. 'OpenFHE: Open-Source Fully Homomorphic Encryption Library'. In: *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 2022, pp. 53–63 (cited on pages 189, 210, 252).

[Alb+18] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan. *Homomorphic Encryption Security Standard*. Tech. rep. Toronto, Canada: HomomorphicEncryption.org, Nov. 2018 (cited on pages 204, 225).

[AP13] J. Alperin-Sheriff and C. Peikert. 'Practical Bootstrapping in Quasilinear Time'. In: *Advances in Cryptology – CRYPTO 2013*. 2013, pp. 1–20 (cited on page 114).

[AP14] J. Alperin-Sheriff and C. Peikert. 'Faster Bootstrapping with Polynomial Error'. In: *Advances in Cryptology – CRYPTO 2014*. 2014, pp. 297–314 (cited on pages 144, 146, 156).

[AP23] A. Al Badawi and Y. Polyakov. *Demystifying Bootstrapping in Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2023/149. 2023 (cited on page 9).

[App+09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. 'Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems'. In: *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*. 2009, pp. 595–618 (cited on pages 66, 67, 156).

[APS15] M. R. Albrecht, R. Player, and S. Scott. 'On the concrete hardness of Learning with Errors'. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203 (cited on page 254).

[AR05] D. Aharonov and O. Regev. 'Lattice problems in NP ∩ coNP'. In: *J. ACM* (Sept. 2005), pp. 749–765 (cited on page 31).

[Bab86] L. Babai. 'On Lovász' lattice reduction and the nearest lattice point problem'. In: *Combinatorica* 6.1 (Mar. 1986), pp. 1–13 (cited on pages 39, 41).

[Bad+20] A. A. Badawi, L. Hoang, C. F. Mun, K. Laine, and K. M. M. Aung. 'PrivFT: Private and Fast Text Classification With Homomorphic Encryption'. In: *IEEE Access* 8 (2020), pp. 226544–226556 (cited on page 210).

[Ban93] W. Banaszczyk. 'New bounds in some transference theorems in the geometry of numbers'. In: *Mathematische Annalen* 296.1 (1993), pp. 625–635 (cited on page 148).

[BDR21] P. Bhargava, A. Drozd, and A. Rogers. 'Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics'. In: *Proceedings of the Second Workshop on Insights from Negative Results in NLP*. Nov. 2021, pp. 125–135 (cited on pages 210, 230).

[Bec97] B. Beckermann. 'On the numerical condition of polynomial bases: estimates for the condition number of Vandermonde, Krylov and Hankel matrices'. PhD thesis. 1997 (cited on page 131).

[Ben+23a] A. Benamira, T. Guérand, T. Peyrin, and S. Saha. *TT-TFHE: a Torus Fully Homomorphic Encryption-Friendly Neural Network Architecture*. 2023 (cited on pages 13, 186–188).

[Ben+23b] H. Bennett, A. Ganju, P. Peetathawatchai, and N. Stephens-Davidowitz. 'Just how hard are rotations of $\mathbb{Z}^n$? Algorithms and cryptography with the simplest lattice'. In: *Advances in Cryptology – EUROCRYPT 2023*. 2023, pp. 252–281 (cited on pages 93, 136).

[Ben23] H. Bennett. 'The Complexity of the Shortest Vector Problem'. In: *SIGACT News* 54.1 (Mar. 2023), pp. 37–61 (cited on page 31).

[BGN05] D. Boneh, E.-J. Goh, and K. Nissim. 'Evaluating 2-DNF Formulas on Ciphertexts'. In: *Theory of Cryptography*. 2005, pp. 325–341 (cited on pages 5, 134).

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. '(Leveled) fully homomorphic encryption without bootstrapping'. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012, pp. 309–325 (cited on pages 9, 12, 98, 104, 105, 112–114, 123, 140, 142, 158, 263).

[BJ66] C. Böhm and G. Jacopini. 'Flow diagrams, turing machines and languages with only two formation rules'. In: *Commun. ACM* 9.5 (May 1966), pp. 366–371 (cited on page 98).

[BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton. *Layer Normalization*. 2016 (cited on page 220).

[Bli14] H. F. Blichfeldt. 'A New Principle in the Geometry of Numbers, with Some Applications'. In: *Transactions of the American Mathematical Society* 15.3 (1914), pp. 227–235 (cited on page 24).

[BLL25] M. F. Bollauf, M. Lie, and C. Ling. 'On Gaussian sampling for q-ary lattices and linear codes with Lee weight'. In: *Annual International Cryptology Conference*. 2025, pp. 321–352 (cited on page 177).

[BMM25] P. Branco, G. Malavolta, and Z. Maradni. *Fully-Homomorphic Encryption from Lattice Isomorphism*. Cryptology ePrint Archive, Paper 2025/993. 2025 (cited on pages 133, 137, 138, 181).

[BNP22] E. Balsa, H. Nissenbaum, and S. Park. 'Cryptography, Trust and Privacy: It's Complicated'. In: *Proceedings of the 2022 Symposium on Computer Science and Law*. 2022, pp. 167–179 (cited on page 258).

[Bon+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. 'Threshold Cryptosystems from Threshold Fully Homomorphic Encryption'. In: *Advances in Cryptology – CRYPTO 2018*. 2018, pp. 565–596 (cited on page 3).

[Bon+22] C. Bonte, I. Iliashenko, J. Park, H. V. Pereira, and N. P. Smart. 'FINAL: faster FHE instantiated with NTRU and LWE'. In: *Advances in Cryptology – ASIACRYPT 2022*. 2022, pp. 188–215 (cited on page 181).

[Bos+16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. 'Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE'. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1006–1018 (cited on page 73).

[Bos+23] J. W. Bos, O. Bronchain, L. Ducas, S. Fehr, Y.-H. Huang, T. Pornin, E. W. Postlethwaite, T. Prest, L. N. Pulles, and W. van Woerden. *HAWK*. Tech. rep. 2023 (cited on pages 48, 259).

[Bou+20] C. Boura, N. Gama, M. Georgieva, and D. Jetchev. 'CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes'. In: *Journal of Mathematical Cryptology* 14.1 (2020), pp. 316–338 (cited on page 134).

[Bou+21] J.-P. Boussat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Habaux. 'Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys'. In: *Advances in Cryptology – EUROCRYPT 2021*. 2021, pp. 587–617 (cited on pages 191, 213).

[BP23] H. Bennett and C. Peikert. 'Hardness of the (Approximate) Shortest Vector Problem: A Simple Proof via Reed-Solomon Codes'. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 275. 2023, 37:1–37:20 (cited on page 31).

[Bra+13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. 'Classical hardness of learning with errors'. In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. 2013, pp. 575–584 (cited on pages 50, 65, 66).

[Bra12] Z. Brakerski. 'Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP'. In: *Advances in Cryptology – CRYPTO 2012*. 2012, pp. 868–886 (cited on pages 114, 263).

[BT06] A. Bogdanov and L. Trevisan. 'Average-Case Complexity'. In: *Foundations and Trends in Theoretical Computer Science* 2.1 (Dec. 2006), pp. 1–106 (cited on page 18).

[BTH22] J.-P. Bossuat, J. Troncoso-Pastoriza, and J.-P. Hubaux. 'Bootstrapping for Approximate Homomorphic Encryption with Negligible Failure-Probability by Using Sparse-Secret Encapsulation'. In: *Applied Cryptography and Network Security: 20th International Conference (ACNS '22)*. 2022, pp. 521–541 (cited on pages 126, 131).

[Buc+14] J. Buchmann, D. Cabarcas, F. Göpfert, A. Hülsing, and P. Weiden. 'Discrete Ziggurat: A Time-Memory Trade-Off for Sampling from a Gaussian Distribution over the Integers'. In: *Selected Areas in Cryptography – SAC 2013*. 2014, pp. 402–417 (cited on page 54).

[BV11] Z. Brakerski and V. Vaikuntanathan. 'Efficient Fully Homomorphic Encryption from (Standard) LWE'. In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. 2011, pp. 97–106 (cited on pages 108, 110, 141, 143, 263).

[Cas+21] L. de Castro, R. Agrawal, R. Yazicigil, A. Chandrakasan, V. Vaikuntanathan, C. Juvekar, and A. Joshi. *Does Fully Homomorphic Encryption Need Compute Acceleration?* 2021 (cited on pages 187, 208).

[CCS19] H. Chen, I. Chillotti, and Y. Song. 'Improved Bootstrapping for Approximate Homomorphic Encryption'. In: *Advances in Cryptology – EUROCRYPT 2019*. 2019, pp. 34–54 (cited on pages 124, 126, 128, 129, 131, 190).

[CDW17] R. Cramer, L. Ducas, and B. Wesolowski. 'Short Stickelberger Class Relations and Application to Ideal-SVP'. In: *Advances in Cryptology – EUROCRYPT 2017*. 2017, pp. 324–348 (cited on pages 88, 90).

[CGGI20] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. 'TFHE: Fast Fully Homomorphic Encryption Over the Torus'. In: *Journal of Cryptology* 33.1 (Jan. 2020), pp. 34–91 (cited on pages 9, 232, 266).

[Che+18a] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. 'A Full RNS Variant of Approximate Homomorphic Encryption'. In: *Selected Areas in Cryptography – SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers*. 2018, pp. 347–368 (cited on page 116).

[Che+18b] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. 'Bootstrapping for Approximate Homomorphic Encryption'. In: *Advances in Cryptology – EUROCRYPT 2018*. 2018, pp. 360–384 (cited on pages 124, 126–128).

[Che+19] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee. 'Numerical Method for Comparison on Homomorphically Encrypted Numbers'. In: *Advances in Cryptology – ASIACRYPT 2019*. 2019, pp. 415–445 (cited on page 233).

[Che+22] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei. 'THE-X: Privacy-Preserving Transformer Inference with Homomorphic Encryption'. In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland, May 2022, pp. 3510–3520 (cited on page 211).

[Che+24a] M. Checri, R. Sirdey, A. Boudguiga, and J.-P. Bultel. 'On the Practical IND-CPA^{D} Security of "exact" and Threshold FHE Schemes and Libraries'. In: *Advances in Cryptology – CRYPTO 2024*. 2024, pp. 3–33 (cited on page 100).

[Che+24b] J. H. Cheon, H. Choe, A. Passelègue, D. Stehlé, and E. Suvanto. 'Attacks Against the IND-CPA^{D} Security of Exact FHE Schemes'. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 2505–2519 (cited on pages 100, 140).

[Che+25] J. H. Cheon, G. Hanrot, J. Kim, and D. Stehlé. *SHIP: A Shallow and Highly Parallelizable CKKS Bootstrapping Algorithm*. Cryptology ePrint Archive, Paper 2025/784. 2025 (cited on pages 126, 131).

[Chu+13] K.-M. Chung, D. Dadush, F.-H. Liu, and C. Peikert. ' On the Lattice Smoothing Parameter Problem '. In: *2013 IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 230–241 (cited on page 52).

[Cin+24] V. Cini, G. Malavolta, N. K. Nguyen, and H. Wee. 'Polynomial Commitments from Lattices: Post-Quantum Security, Fast Verification and Transparent Setup'. In: *Advances in Cryptology – CRYPTO 2024*. 2024, pp. 207–242 (cited on page 55).

[CKK20] J. H. Cheon, D. Kim, and D. Kim. 'Efficient Homomorphic Comparison Methods with Optimal Complexity'. In: *Advances in Cryptology – ASIACRYPT 2020*. 2020, pp. 221–256 (cited on pages 233, 242).

[CKKS17] J. H. Cheon, A. Kim, M. Kim, and Y. Song. 'Homomorphic Encryption for Arithmetic of Approximate Numbers'. In: *Advances in Cryptology – ASIACRYPT 2017*. 2017, pp. 409–437 (cited on pages 13, 116, 121, 188, 210, 213, 232).

[CKS13] A. Chatterjee, M. Kaushal, and I. Sengupta. 'Accelerating Sorting of Fully Homomorphic Encrypted Data'. In: *Progress in Cryptology – INDOCRYPT 2013*. 2013, pp. 262–273 (cited on page 232).

[CKY25] J. H. Cheon, J. Kim, and Y. Yeo. *OverModRaise: Reducing Modulus Consumption of CKKS Bootstrapping*. Cryptology ePrint Archive, Paper 2025/1298. 2025 (cited on pages 126, 131).

[CLW25] V. Cini, R. W. F. Lai, and I. K. Y. Woo. 'Lattice-Based Obfuscation from NTRU and Equivocal LWE'. In: *Advances in Cryptology – CRYPTO 2025*. 2025, pp. 39–72 (cited on page 55).

[Col+23] S. Colombo, K. Nikitin, H. Corrigan-Gibbs, D. J. Wu, and B. Ford. 'Authenticated private information retrieval'. In: *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Aug. 2023, pp. 3835–3851 (cited on page 11).

[Con+25] K. Cong, R. Geelen, J. Kang, and J. Park. 'Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification'. In: *Selected Areas in Cryptography – SAC 2024: 31st International Conference, Montreal, QC, Canada, August 28–30, 2024, Revised Selected Papers, Part I*. 2025, pp. 3–25 (cited on page 232).

[Cop97] D. Coppersmith. 'Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities'. In: *Journal of Cryptology* 10.4 (Sept. 1997), pp. 233–260 (cited on pages 10, 19).

[CP23] L. de Castro and C. Peikert. 'Functional Commitments for All Functions, with Transparent Setup and from SIS'. In: *Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III*. 2023, pp. 287–320 (cited on page 62).

[CR88] B. Chor and R. Rivest. 'A knapsack-type public key cryptosystem based on arithmetic in finite fields'. In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 901–909 (cited on page 18).

[Cra+16] R. Cramer, L. Ducas, C. Peikert, and O. Regev. 'Recovering Short Generators of Principal Ideals in Cyclotomic Rings'. In: *Advances in Cryptology – EUROCRYPT 2016*. 2016, pp. 559–585 (cited on pages 88, 90).

[CS13] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*. Vol. 290. Springer Science & Business Media, 2013 (cited on page 177).

[CSZ21] P.-E. Clet, O. Stan, and M. Zuber. 'BFV, CKKS, TFHE: Which One is the Best for a Secure Neural Network Evaluation in the Cloud?' In: *Applied Cryptography and Network Security Workshops*. 2021, pp. 279–300 (cited on page 210).

[Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. June 2019, pp. 4171–4186 (cited on pages 209, 263).

[DH76] W. Diffie and M. Hellman. 'New directions in cryptography'. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654 (cited on page 1).

[Dif88] W. Diffie. 'The first ten years of public-key cryptography'. In: *Proceedings of the IEEE* 76.5 (1988), pp. 560–577 (cited on page 17).

[DLP14a] L. Ducas, V. Lyubashevsky, and T. Prest. 'Efficient Identity-Based Encryption over NTRU Lattices'. In: *Advances in Cryptology – ASIACRYPT 2014*. 2014, pp. 22–41 (cited on page 54).

[DLP14b] L. Ducas, V. Lyubashevsky, and T. Prest. 'Efficient identity-based encryption over NTRU lattices'. In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2014, pp. 22–41 (cited on page 55).

[DM15] L. Ducas and D. Micciancio. 'FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second'. In: *Advances in Cryptology – EUROCRYPT 2015*. 2015, pp. 617–640 (cited on pages 8, 9, 156, 264).

[DPS25] L. Ducas, L. N. Pulles, and M. Stevens. *Towards a Modern LLL Implementation*. Cryptology ePrint Archive, Paper 2025/774. 2025 (cited on page 39).

[Duc+13] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. 'Lattice Signatures and Bimodal Gaussians'. In: *Advances in Cryptology – CRYPTO 2013*. 2013, pp. 40–56 (cited on page 54).

[Duc+22] L. Ducas, E. W. Postlethwaite, L. N. Pulles, and W. v. Woerden. 'Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple'. In: *Advances in Cryptology – ASIACRYPT 2022*. 2022, pp. 65–94 (cited on pages 136, 175).

[DvW22] L. Ducas and W. van Woerden. 'On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography'. In: *Advances in Cryptology – EUROCRYPT 2022*. 2022, pp. 643–673 (cited on pages 13, 92, 93, 95, 96, 136, 137, 175, 176, 181, 182).

[DW21] L. Ducas and W. van Woerden. 'NTRU fatigue: how stretched is overstretched?' In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2021, pp. 3–32 (cited on page 178).

[EC16a] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). OJ L 119, 4.5.2016, p. 1–88, May 4, 2016 (cited on pages 185, 264).

[EC16b] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). OJ L 119, 4.5.2016, p. 1–88, May 4, 2016 (cited on page 258).

[EJK20] T. Espitau, A. Joux, and N. Kharchenko. 'On a Dual/Hybrid Approach to Small Secret LWE'. In: *Progress in Cryptology – INDOCRYPT 2020*. 2020, pp. 440–462 (cited on page 177).

[ElG85] T. ElGamal. 'A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms'. In: *Advances in Cryptology – CRYPTO '84*. 1985, pp. 10–18 (cited on pages 5, 134).

[Emm+15] N. Emmadi, P. Gauravaram, H. Narumanchi, and H. Syed. 'Updates on Sorting of Fully Homomorphic Encrypted Data'. In: *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*. 2015, pp. 19–24 (cited on page 232).

[Eur22] European Commission. *Proposal for a Regulation of the European Parliament and of the Council laying down rules to prevent and combat child sexual abuse (COM(2022) 209 final / 2022/0155(COD))*. 2022. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52022PC0209 (cited on page 258).

[EWY23] T. Espitau, A. Wallet, and Y. Yu. 'On Gaussian sampling, smoothing parameter and application to signatures'. In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2023, pp. 65–97 (cited on page 177).

[EZ66] H. Ehlich and K. Zeller. 'Auswertung der Normen von Interpolationsoperatoren'. In: *Mathematische Annalen* 164.2 (June 1966), pp. 105–112 (cited on page 130).

[Fou+18] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang, et al. 'Falcon: Fast-Fourier lattice-based compact signatures over NTRU'. In: *Submission to the NIST's post-quantum cryptography standardization process* 36.5 (2018), pp. 1–75 (cited on page 48).

[Fro+21] D. Froelicher, J. R. Troncoso-Pastoriza, J. L. Raisaro, M. A. Cuendet, J. S. Sousa, H. Cho, B. Berger, J. Fellay, and J.-P. Hubaux. 'Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption'. In: *Nature Communications* 12.1 (Oct. 2021), p. 5910 (cited on page 11).

[FV12] J. Fan and F. Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. 2012 (cited on pages 114, 263).

[Gal12] S. D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012 (cited on page 34).

[Gam+16] N. Gama, M. Izabachene, P. Q. Nguyen, and X. Xie. 'Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems'. In: *Advances in Cryptology–EUROCRYPT 2016*. 2016, pp. 528–558 (cited on pages 155, 156).

[Gen+20] N. Genise, D. Micciancio, C. Peikert, and M. Walter. 'Improved Discrete Gaussian and Subgaussian Analysis for Lattice Cryptography'. In: *Public-Key Cryptography – PKC 2020*. 2020 (cited on page 54).

[Gen09a] C. Gentry. 'A fully homomorphic encryption scheme'. Ph.D. Thesis, Stanford University. PhD thesis. Stanford University, 2009 (cited on page 8).

[Gen09b] C. Gentry. 'Fully homomorphic encryption using ideal lattices'. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing – STOC '09*. 2009, pp. 169–178 (cited on pages 8, 99).

[GG98] O. Goldreich and S. Goldwasser. 'On the limits of non-approximability of lattice problems'. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. 1998, pp. 1–9 (cited on page 31).

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. 'Public-key cryptosystems from lattice reduction problems'. In: *Advances in Cryptology — CRYPTO '97*. 1997, pp. 112–131 (cited on page 48).

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. 'Fully Homomorphic Encryption with Polylog Overhead'. In: *Advances in Cryptology – EUROCRYPT 2012*. 2012, pp. 465–482 (cited on pages 121, 123).

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. 'Homomorphic Evaluation of the AES Circuit'. In: *Advances in Cryptology – CRYPTO 2012*. 2012 (cited on pages 104, 111, 112, 121, 140, 143, 264).

[Gil+16] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. 'CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy'. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. New York, New York, USA: PMLR, June 2016, pp. 201–210 (cited on page 185).

[GM82] S. Goldwasser and S. Micali. 'Probabilistic encryption & how to play mental poker keeping secret all partial information'. In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. 1982, pp. 365–377 (cited on pages 5, 134).

[GMP19] N. Genise, D. Micciancio, and Y. Polyakov. 'Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More'. In: *Advances in Cryptology – EUROCRYPT 2019*. 2019, pp. 655–684 (cited on page 145).

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. 'How to play ANY mental game'. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. 1987, pp. 218–229 (cited on page 2).

[GN08] N. Gama and P. Nguyen. 'Predicting Lattice Reduction'. In: *Advances in Cryptology – EUROCRYPT 2008*. 2008, pp. 31–51 (cited on page 31).

[GNR10] N. Gama, P. Q. Nguyen, and O. Regev. 'Lattice Enumeration Using Extreme Pruning'. In: *Advances in Cryptology – EUROCRYPT 2010*. 2010, pp. 257–278 (cited on page 62).

[Gol+13] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. 'How to Run Turing Machines on Encrypted Data'. In: *Advances in Cryptology – CRYPTO 2013*. 2013, pp. 536–553 (cited on pages 98, 231).

[Gol+99] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. 'Approximating shortest lattice vectors is not harder than approximating closest lattice vectors'. In: *Information Processing Letters* 71.2 (1999), pp. 55–61 (cited on page 32).

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. 'Trapdoors for hard lattices and new cryptographic constructions'. In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. 2008, pp. 197–206 (cited on pages 47, 54, 62).

[GSW13] C. Gentry, A. Sahai, and B. Waters. 'Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based'. In: *Advances in Cryptology – CRYPTO 2013*. 2013, pp. 75–92 (cited on pages 12, 144, 158).

[GV23] R. Geelen and F. Vercauteren. 'Bootstrapping for BGV and BFV Revisited'. In: *Journal of Cryptology* 36.2 (Mar. 2023), p. 12 (cited on page 115).

[GV25] R. Geelen and F. Vercauteren. 'Fully Homomorphic Encryption for Cyclotomic Prime Moduli'. In: *Advances in Cryptology – EUROCRYPT 2025*. 2025, pp. 366–397 (cited on page 114).

[He+16a] K. He, X. Zhang, S. Ren, and J. Sun. 'Deep Residual Learning for Image Recognition'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778 (cited on pages 13, 221).

[He+16b] K. He, X. Zhang, S. Ren, and J. Sun. 'Deep Residual Learning for Image Recognition'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cited on page 265).

[HG23] D. Hendrycks and K. Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023 (cited on page 222).

[Hof+03] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. 'NTRUSign: Digital Signatures Using the NTRU Lattice'. In: *Topics in Cryptology — CT-RSA 2003*. 2003, pp. 122–140 (cited on page 48).

[Hon+21] S. Hong, S. Kim, J. Choi, Y. Lee, and J. H. Cheon. 'Efficient Sorting of Homomorphic Encrypted Data With k-Way Sorting Network'. In: *IEEE Trans. on Information Forensics and Security* 16 (2021), pp. 4389–4404 (cited on pages 15, 232, 235, 239, 256, 257).

[Hon+22] S. Hong, J. H. Park, W. Cho, H. Choe, and J. H. Cheon. 'Secure tumor classification by shallow neural network using homomorphic encryption'. In: *BMC Genomics* 23.1 (Apr. 2022), p. 284 (cited on page 211).

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. 'NTRU: A ring-based public key cryptosystem'. In: *Algorithmic Number Theory*. 1998, pp. 267–288 (cited on page 74).

[HR14] I. Haviv and O. Regev. 'On the Lattice Isomorphism Problem'. In: *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2014, pp. 391–404 (cited on page 92).

[HS14] S. Halevi and V. Shoup. 'Algorithms in HElib'. In: *Advances in Cryptology – CRYPTO 2014*. 2014, pp. 554–571 (cited on pages 129, 190, 214).

[Hua+22] H. Huang, Y. Wang, L. Wang, H. Ge, and Q. Gu. 'Secure word-level sorting based on fully homomorphic encryption'. In: *J. Inf. Secur. Appl.* 71 (2022) (cited on page 235).

[Imp95] R. Impagliazzo. 'A personal view of average-case complexity'. In: *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*. 1995, pp. 134–147 (cited on page 69).

[IS15] S. Ioffe and C. Szegedy. 'Batch normalization: accelerating deep network training by reducing internal covariate shift'. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. 2015, pp. 448–456 (cited on page 190).

[Jou+17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 'Bag of Tricks for Efficient Text Classification'. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain, Apr. 2017, pp. 427–431 (cited on page 210).

[JVC18a] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. 'GAZELLE: A Low Latency Framework for Secure Neural Network Inference'. In: *27th USENIX Security Symposium*. Aug. 2018, pp. 1651–1669 (cited on pages 186, 194).

[JVC18b] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. 'GAZELLE: A Low Latency Framework for Secure Neural Network Inference'. In: *Proceedings of the 27th USENIX Conference on Security Symposium*. 2018 (cited on page 214).

[KF17] P. Kirchner and P.-A. Fouque. 'Revisiting lattice attacks on overstretched NTRU parameters'. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 2017, pp. 3–26 (cited on page 178).

[KG23a] D. Kim and C. Guyot. 'Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption'. In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 2175–2187 (cited on pages 13, 14, 187).

[KG23b] D. Kim and C. Guyot. 'Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption'. In: *IEEE Transactions on Information Forensics and Security* 18 (Mar. 2023), pp. 2175–2187 (cited on pages 186–188, 206, 208).

[KH09] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. 0. Toronto, Ontario: University of Toronto, 2009 (cited on page 186).

[Kho04] S. Khot. 'Hardness of approximating the shortest vector problem in lattices'. In: *45th Annual IEEE Symposium on Foundations of Computer Science*. 2004, pp. 126–135 (cited on page 33).

[Kim+22a] S. Kim, M. Park, J. Kim, T. Kim, and C. Min. 'EvalRound Algorithm in CKKS Bootstrapping'. In: *Advances in Cryptology – ASIACRYPT 2022*. 2022, pp. 161–187 (cited on page 126).

[Kim+22b] S. Kim, M. Park, J. Kim, T. Kim, and C. Min. 'EvalRound algorithm in CKKS bootstrapping'. In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2022, pp. 161–187 (cited on page 131).

[Kim+25] S. Kim, E. Ünay, A. Yilmazer-Metin, and H. T. Lee. *Optimized Rank Sort for Encrypted Real Numbers*. Cryptology ePrint Archive, Paper 2025/1170. 2025 (cited on pages 234, 254).

[Kle00] P. Klein. 'Finding the closest lattice vector when it's unusually close'. In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. 2000, pp. 937–941 (cited on page 54).

[KMS25] P. Kern, E. Manino, and C. Sinz. 'Certified Error Analysis of Homomorphically Encrypted Neural Networks'. In: *AI Verification, International Symposium, SAIV 2025*. 2025 (cited on page 260).

[Ko25] R. Ko. *The Beginner's Textbook for Fully Homomorphic Encryption*. 2025 (cited on page 104).

[KPP22] A. Kim, A. Papadimitriou, and Y. Polyakov. 'Approximate Homomorphic Encryption with Reduced Approximation Error'. In: *Topics in Cryptology – CT-RSA 2022*. 2022, pp. 120–144 (cited on pages 116, 123, 189, 210, 225, 226, 230, 252).

[KPZ21] A. Kim, Y. Polyakov, and V. Zucca. 'Revisiting Homomorphic Encryption Schemes for Finite Fields'. In: *Advances in Cryptology – ASIACRYPT 2021*. 2021, pp. 608–639 (cited on pages 105, 110–112, 126, 253).

[KTX08] A. Kawachi, K. Tanaka, and K. Xagawa. 'Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems'. In: *Advances in Cryptology - ASIACRYPT 2008*. 2008, pp. 372–389 (cited on page 62).

[KY20] J. Kats and L. Yehuda. *Introduction to Modern Cryptography (3rd ed.)* Chapman and Hall, 2020 (cited on page 1).

[KZ73] A. Korkine and G. Zolotareff. 'Sur les formes quadratiques'. In: *Mathematische Annalen* 6.3 (Sept. 1873), pp. 366–389 (cited on page 43).

[Lat+23] B. S. Latibari, K. I. Gubbi, H. Homayoun, and A. Sasan. 'A Survey on FHE Acceleration'. In: *2023 IEEE 16th Dallas Circuits and Systems Conference (DCAS)*. 2023, pp. 1–6 (cited on page 208).

[Lat24] *Lattigo v6*. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. Aug. 2024 (cited on page 206).

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. 'Deep learning'. In: *Nature* 521.7553 (May 2015), pp. 436–444 (cited on page 184).

[Le +90] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. 'Handwritten digit recognition with a back-propagation network'. In: *Advances in Neural Information Processing Systems 2*. 1990, pp. 396–404 (cited on pages 185, 190).

[Lee+21] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No. 'High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Polynomial Approximation and Inverse Sine Function'. In: *Advances in Cryptology – EUROCRYPT 2021*. 2021, pp. 618–647 (cited on page 126).

[Lee+22a] E. Lee, J.-W. Lee, Y.-S. Kim, and J.-S. No. 'Optimization of Homomorphic Comparison Algorithm on RNS-CKKS Scheme'. In: *IEEE Access* 10 (2022), pp. 26163–26176 (cited on page 235).

[Lee+22b] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and W. Choi. 'Low-Complexity Deep Convolutional Neural Networks on Fully Homomorphic Encryption Using Multiplexed Parallel Convolutions'. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. 2022, pp. 12403–12422 (cited on pages 186–188, 198, 208).

[Lee+22c] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim. 'Minimax Approximation of Sign Function by Composite Polynomial for Homomorphic Comparison'. In: *IEEE Transactions on Dependable and Secure Computing* 19.6 (2022), pp. 3711–3727 (cited on pages 233, 235, 248).

[Lee+22d] G. Lee, M. Kim, J. H. Park, S.-w. Hwang, and J. H. Cheon. 'Privacy-Preserving Text Classification on BERT Embeddings with Homomorphic Encryption'. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. July 2022, pp. 3169–3175 (cited on page 211).

[Lee22] J.-W. e. a. Lee. 'Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network'. In: *IEEE Access* 10 (2022), pp. 30039–30054 (cited on pages 13, 14, 187, 188, 211).

[Li+24] B. Li, D. Micciancio, M. Raykova, and M. Schultz-Wu. 'Hintless Single-Server Private Information Retrieval'. In: *Advances in Cryptology – CRYPTO 2024*. 2024, pp. 183–217 (cited on page 11).

[Lin20] Y. Lindell. 'Secure multiparty computation'. In: *Communications of the ACM* 64.1 (2020), pp. 86–96 (cited on page 3).

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. 'Factoring polynomials with rational coefficients'. In: *Mathematische Annalen* 261.4 (Dec. 1982), pp. 515–534 (cited on pages 34, 37).

[LM06] V. Lyubashevsky and D. Micciancio. 'Generalized compact knapsacks are collision resistant'. In: *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*. Venice, Italy, 2006, pp. 144–155 (cited on pages 74, 79, 90).

[LM09] V. Lyubashevsky and D. Micciancio. 'On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem'. In: *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*. 2009, pp. 577–594 (cited on page 33).

[LM21] B. Li and D. Micciancio. 'On the Security of Homomorphic Encryption on Approximate Numbers'. In: *Advances in Cryptology – EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*. 2021, pp. 648–677 (cited on pages 100, 139).

[LMP23] Z. Liu, D. Micciancio, and Y. Polyakov. 'Large-Precision Homomorphic Sign Evaluation Using FHEW/TFHE Bootstrapping'. In: *Advances in Cryptology - ASIACRYPT 2022*. 2023, pp. 130–160 (cited on page 234).

[LNP22] V. Lyubashevsky, N. K. Nguyen, and M. Plançon. 'Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General'. In: *Advances in Cryptology – CRYPTO 2022*. 2022, pp. 71–101 (cited on page 55).

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. 'On Ideal Lattices and Learning with Errors over Rings'. In: *Advances in Cryptology – EUROCRYPT 2010*. 2010, pp. 1–23 (cited on pages 91, 92).

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. 'A Toolkit for Ring-LWE Cryptography'. In: *Advances in Cryptology – EUROCRYPT 2013*. 2013, pp. 35–54 (cited on pages 118, 119).

[LRvW25] A. Leporati, L. Rovida, and W. van Woerden. 'Beyond LWE: a Lattice Framework for Homomorphic Encryption'. In: *IACR ePrint Archive 2025/1171* (2025) (cited on pages 10, 12).

[Lu+21] W.-j. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu. 'PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption'. In: *2021 IEEE Symposium on Security and Privacy (S&P)*. 2021, pp. 1057–1073 (cited on pages 232, 235).

[Lyu12] V. Lyubashevsky. 'Lattice Signatures without Trapdoors'. In: *Advances in Cryptology – EUROCRYPT 2012*. 2012, pp. 738–755 (cited on page 63).

[Lyu24] V. Lyubashevsky. *Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA)*. Cryptology ePrint Archive, Paper 2024/1287. 2024 (cited on pages 61, 69).

[Mar18] D. Marcus. *Number Fields*. Springer International Publishing, 2018 (cited on page 73).

[MCR21] M. H. Mughees, H. Chen, and L. Ren. 'OnionPIR: Response Efficient Single-Server PIR'. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 2292–2306 (cited on page 11).

[MEHP25] F. Mazzone, M. Everts, F. Hahn, and A. Peter. 'Efficient Ranking, Order Statistics, and Sorting under CKKS'. In: *34th USENIX Security Symposium (USENIX Security '25)*. Aug. 2025 (cited on pages 15, 232, 234, 236, 238, 241, 242, 245–247, 253, 254, 256, 257).

[Mer78] R. C. Merkle. 'Secure communications over insecure channels'. In: *Commun. ACM* 21.4 (1978), pp. 294–299 (cited on page 1).

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*. Vol. 671. Boston, Massachusetts, Mar. 2002 (cited on pages 19, 24, 33).

[MH78] R. Merkle and M. Hellman. 'Hiding information and signatures in trapdoor knapsacks'. In: *IEEE Transactions on Information Theory* 24.5 (1978), pp. 525–530 (cited on page 18).

[Mic01a] D. Micciancio. 'The hardness of the closest vector problem with preprocessing'. In: *IEEE Transactions on Information Theory* 47.3 (2001), pp. 1212–1215 (cited on page 33).

[Mic01b] D. Micciancio. 'Improving Lattice Based Cryptosystems Using the Hermite Normal Form'. In: *Cryptography and Lattices*. 2001, pp. 126–145 (cited on page 66).

[Mic02] D. Micciancio. 'Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions'. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 2002, pp. 356–365 (cited on page 74).

[Mic12] D. Micciancio. *Lecture 3: The LLL Algorithm*. Course Lecture Notes for CSE 206A: Lattice Algorithms and Applications. Course Lecture Notes for CSE 206A: Lattice Algorithms and Applications, UCSD. 2012. URL: https://cseweb.ucsd.edu/classes/wi12/cse206A-a/lec3.pdf (cited on page 37).

[Mic21] D. Micciancio. *CSE 206A: Lattice Algorithms and Applications*. Course Lecture Notes, UCSD. 2021. URL: https://cseweb.ucsd.edu/classes/fa21/cse206A-a/ (cited on page 19).

[Mic98] D. Micciancio. 'The shortest vector in a lattice is hard to approximate to within some constant'. In: *Proceedings 39th Annual Symposium on Foundations of Computer Science*. 1998, pp. 92–98 (cited on pages 31, 33).

[Min96] H. Minkowski. *Geometrie der Zahlen*. Leipzig: Teubner, 1896 (cited on page 23).

[ML25] C. M and B. L. 'Perceptual Hashing on Images Encrypted with a Homomorphic Scheme'. In: Publications Office of the European Union (online) (2025). DOI: 10.2760/2947924 (cited on page 11).

[MLS25] S. Min, J.-W. Lee, and Y. Song. 'Enhanced CKKS Bootstrapping with Generalized Polynomial Composites Approximation'. In: *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*. 2025, pp. 1–12 (cited on pages 126, 131).

[MM11] D. Micciancio and P. Mol. 'Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions'. In: *Advances in Cryptology – CRYPTO 2011*. 2011, pp. 465–484 (cited on page 65).

[Mon+23] J. Mono, C. Marcolla, G. Land, T. Güneysu, and N. Aaraj. 'Finding and Evaluating Parameters for BGV'. In: *Progress in Cryptology - AFRICACRYPT 2023*. 2023, pp. 370–394 (cited on pages 114, 163).

[Mon+24] L. Montero, J. Frery, C. Kherfallah, R. Bredehoft, and A. Stoian. 'Machine Learning Training on Encrypted Data with TFHE'. In: *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics*. 2024, pp. 71–76 (cited on page 11).

[MP12] D. Micciancio and C. Peikert. 'Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller'. In: *Advances in Cryptology – EUROCRYPT 2012*. 2012, pp. 700–718 (cited on pages 47, 54, 145).

[MR07] D. Micciancio and O. Regev. 'Worst-Case to Average-Case Reductions Based on Gaussian Measures'. In: *SIAM Journal on Computing* 37.1 (2007), pp. 267–302 (cited on pages 52, 54).

[Muk+23] A. Mukherjee, A. Aikata, A. C. Mert, Y. Lee, S. Kwon, M. Deryabin, and S. Sinha Roy. 'ModHE: Modular Homomorphic Encryption Using Module Lattices: Potentials and Limitations'. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2024.1 (Dec. 2023), pp. 527–562 (cited on page 88).

[MV24] D. Micciancio and V. Vaikuntanathan. 'SoK: Learning with Errors, Circular Security, and Fully Homomorphic Encryption'. In: *Public-Key Cryptography – PKC 2024*. 2024, pp. 291–321 (cited on pages 8, 110).

[MW17] D. Micciancio and M. Walter. 'Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time'. In: *Advances in Cryptology – CRYPTO 2017*. 2017, pp. 455–485 (cited on page 54).

[MW18] D. Micciancio and M. Walter. 'On the Bit Security of Cryptographic Primitives'. In: *Advances in Cryptology – EUROCRYPT 2018*. 2018, pp. 3–28 (cited on page 17).

[MW22] S. J. Menon and D. J. Wu. 'SPIRAL: Fast, High-Rate Single-Server PIR via FHE Composition'. In: *2022 IEEE Symposium on Security and Privacy (SP)*. 2022, pp. 930–947 (cited on page 11).

[Ngu+25] L. Nguyen, Q. B. Phan, L. Zhang, and T. T. Nguyen. 'A Novel Polynomial Activation for Audio Classification Using Homomorphic Encryption'. In: *IEEE Access* 13 (2025), pp. 87834–87847 (cited on page 11).

[Nju+25] N. B. Njungle, E. Jahns, L. Mastromauro, E. P. Kayang, M. Stojkov, and M. A. Kinsy. *Prismo: A Decision Support System for Privacy-Preserving ML Framework Selection*. arXiv paper 2510.09985. 2025 (cited on page 14).

[Nol+24] N. Nolte, M. Malhou, E. Wenger, S. Stevens, C. Li, F. Charton, and K. Lauter. 'The Cool and the Cruel: Separating Hard Parts of LWE Secrets'. In: *Progress in Cryptology - AFRICACRYPT 2024: 15th International Conference on Cryptology in Africa, Douala, Cameroon, July 10–12, 2024, Proceedings*. 2024, pp. 428–453 (cited on page 177).

[NR06] P. Q. Nguyen and O. Regev. 'Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures'. In: *Advances in Cryptology - EUROCRYPT 2006*. 2006, pp. 271–288 (cited on pages 47, 48).

[NV09] P. Q. Nguyen and B. Valle. *The LLL Algorithm: Survey and Applications*. 1st. Springer Publishing Company, Incorporated, 2009 (cited on page 39).

[Odl90] A. M. Odlyzko. 'The Rise and Fall of Knapsack Cryptosystems'. In: *Cryptology and Computational Number Theory*. Vol. 42. 1990, pp. 75–88 (cited on page 19).

[Pai99] P. Paillier. 'Public-Key Cryptosystems Based on Composite Degree Residuosity Classes'. In: *Advances in Cryptology — EUROCRYPT '99*. 1999, pp. 223–238 (cited on pages 5, 134).

[Pei09] C. Peikert. 'Public-key cryptosystems from the worst-case shortest vector problem: extended abstract'. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. 2009, pp. 333–342 (cited on page 65).

[Pei10] C. Peikert. 'An Efficient and Parallel Gaussian Sampler for Lattices'. In: *Advances in Cryptology – CRYPTO 2010*. 2010, pp. 80–97 (cited on page 54).

[Pei13] C. Peikert. *Lattice-based cryptography: Short Integer Solution (SIS) and Learning With Errors (LWE)*. Accessed: 2025-09-01. 2013. URL: https://cosec.bit.uni-bonn.de/students/events/cryptabit2013 (cited on pages 60, 69).

[Pei22] C. Peikert. *Lattices in Cryptography*. https://github.com/cpeikert/LatticesInCryptography. Univesrity of Michigan, Course notes. 2022 (cited on page 19).

[PHS19] A. Pellet-Mary, G. Hanrot, and D. Stehlé. 'Approx-SVP in Ideal Lattices with Pre-processing'. In: *Advances in Cryptology – EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II*. 2019, pp. 685–716 (cited on pages 88, 90).

[PL05] B. Pang and L. Lee. 'Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales'. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. June 2005, pp. 115–124 (cited on page 212).

[Pol71] J. M. Pollard. 'The Fast Fourier Transform in a Finite Field'. In: *Mathematics of Computation* 25.114 (1971), pp. 365–374 (cited on page 88).

[PP19] C. Peikert and Z. Pepin. 'Algebraically Structured LWE, Revisited'. In: (2019), pp. 1–23 (cited on page 92).

[PR06] C. Peikert and A. Rosen. 'Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices'. In: *Proceedings of the Third Conference on Theory of Cryptography*. 2006, pp. 145–166 (cited on pages 62, 74, 78, 90).

[PRS17] C. Peikert, O. Regev, and N. Stephens-Davidowitz. 'Pseudorandomness of ring-LWE for any ring and modulus'. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 461–473 (cited on page 92).

[PS73] M. S. Paterson and L. J. Stockmeyer. 'On the number of nonscalar multiplications necessary to evaluate polynomials'. In: *SIAM Journal on Computing* 2.1 (1973), pp. 60–66 (cited on pages 131, 190, 227).

[RAD78] R. L. Rivest, L. Adleman, and M. L. Dertouzos. 'On data banks and privacy homomorphisms'. In: *Foundations of secure computation* 4.11 (1978), pp. 169–180 (cited on pages 3, 5, 17, 133).

[Reg05] O. Regev. 'On lattices, learning with errors, random linear codes, and cryptography'. In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. 2005, pp. 84–93 (cited on pages 10, 64, 65, 68, 69, 71, 134, 143).

[RL24a] L. Rovida and A. Leporati. 'Encrypted Image Classification with Low Memory Footprint Using Fully Homomorphic Encryption'. In: *International Journal of Neural Systems* 34.05 (2024), p. 2450025 (cited on pages 12, 13).

[RL24b] L. Rovida and A. Leporati. 'Transformer-based Language Models and Homomorphic Encryption: An Intersection with BERT-tiny'. In: *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics*. 2024, pp. 3–13 (cited on pages 12, 14).

[RLB25] L. Rovida, A. Leporati, and S. Basile. 'Lightweight Sorting in Approximate Homomorphic Encryption'. In: *IACR ePrint Archive 2025/1150* (2025) (cited on pages 12, 15).

[Rot11] R. Rothblum. 'Homomorphic Encryption: From Private-Key to Public-Key'. In: *Theory of Cryptography*. 2011, pp. 219–234 (cited on pages 6, 137).

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. 'A method for obtaining digital signatures and public-key cryptosystems'. In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126 (cited on pages 4, 266).

[Sam+22] N. Samardzic, A. Feldmann, A. Krastev, N. Manohar, N. Genise, S. Devadas, K. Eldefrawy, C. Peikert, and D. Sanchez. 'CraterLake: A Hardware Accelerator for Efficient Unbounded Computation on Encrypted Data'. In: *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 2022 (cited on page 208).

[Sat+23] A. Satriawan, I. Syafalni, R. Mareta, I. Anshori, W. Shalannanda, and A. Barra. 'Conceptual Review on Number Theoretic Transform and Comprehensive Review on Its Implementations'. In: *IEEE Access* 11 (2023), pp. 70288–70316 (cited on page 104).

[Sch87] C. Schnorr. 'A hierarchy of polynomial time lattice basis reduction algorithms'. In: *Theoretical Computer Science* 53.2 (June 1987), pp. 201–224 (cited on page 43).

[SE24] B. C. Stahl and D. Eke. 'The ethics of ChatGPT–Exploring the ethical issues of an emerging technology'. In: *International Journal of Information Management* 74 (2024), p. 102700 (cited on page 260).

[Sei18] G. Seiler. *Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography*. Cryptology ePrint Archive, Paper 2018/039. 2018 (cited on page 104).

[Sha84] A. Shamir. 'A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem'. In: *IEEE Transactions on Information Theory* 30.5 (1984), pp. 699–704 (cited on pages 10, 18, 19).

[SM24] C. Shepherd and K. Markantonakis. *Trusted Execution Environments*. Springer, 2024 (cited on page 2).

[SM25] S. Sun and E. Makri. *SoK: Multiparty Computation in the Preprocessing Model*. Cryptology ePrint Archive, Paper 2025/060. 2025 (cited on page 3).

[Ste04] W. Stein. *A Brief Introduction to Classical and Adelic Algebraic Number Theory*. William Stein, 2004 (cited on page 73).

[Sud+23] K. Sudo, M. Tezuka, K. Hara, and Y. Yoshida. 'Quantum Search-to-Decision Reduction for the LWE Problem'. In: *Progress in Cryptology - AFRICACRYPT 2023*. 2023, pp. 395–413 (cited on page 65).

[SV10] N. P. Smart and F. Vercauteren. 'Fully homomorphic encryption with relatively small key and ciphertext sizes'. In: *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*. 2010, pp. 420–443 (cited on page 115).

[SV14] N. P. Smart and F. Vercauteren. 'Fully homomorphic SIMD operations'. In: *Design, Codes and Cryptography* 71.1 (Apr. 2014), pp. 57–81 (cited on page 115).

[Tre19] L. N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. Society for Industrial and Applied Mathematics, 2019 (cited on pages 130, 211).

[Tur+20] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova. 'Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation'. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020 (cited on page 210).

[Vai15] V. Vaikuntanathan. *Lecture 10: Complexity of Lattice Problems: NP-hardness of (approximate) SVP, and open problems*. Accessed: 2025-09-01. 2015. URL: https://people.csail.mit.edu/vinodv/6876-Fall2015/L10.pdf (cited on page 33).

[Vai21] V. Vaikuntanathan. *6.876J Advanced Topics in Cryptography: Lattices*. Course Lecture Notes, MIT. 2021. URL: https://people.csail.mit.edu/vinodv/6876-Fall2015/index.html (cited on page 19).

[Vas+17a] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 'Attention is All you Need'. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017 (cited on page 14).

[Vas+17b] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 'Attention is All You Need'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6000–6010 (cited on page 209).

[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. 'Fully Homomorphic Encryption over the Integers'. In: *Advances in Cryptology – EUROCRYPT 2010*. 2010, pp. 24–43 (cited on pages 6, 263).

[vEB81] P. van Emde Boas. 'Another NP-complete problem and the complexity of computing short vectors in a lattice'. In: *Tecnical Report, Department of Mathmatics, University of Amsterdam* (1981) (cited on page 33).

[vW23] W. van Woerden. 'Lattice cryptography: from cryptanalysis to New Foundations'. Ph.D. Thesis, Leiden University. PhD thesis. Leiden University, 2023 (cited on page 34).

[vW24] W. van Woerden. 'Dense and smooth lattices in any genus'. In: *Advances in Cryptology – ASIACRYPT 2024*. 2024, pp. 386–417 (cited on pages 137, 181, 182).

[Wil51] M. Wilkes. *The Preparation of Programs for an Electronic Digital Computer: With Special Reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley Press, 1951, pp. 323–324 (cited on page 233).

[WNK20] A. Wood, K. Najarian, and D. Kahrobaei. 'Homomorphic Encryption for Machine Learning in Medicine and Bioinformatics'. In: *ACM Comput. Surv.* 53.4 (Aug. 2020) (cited on page 11).

[Yao86] A. C.-C. Yao. 'How to generate and exchange secrets'. In: *27th Annual Symposium on Foundations of Computer Sciencs*. 1986, pp. 162–167 (cited on page 2).

[ZLJ25] M. Zheng, Q. Lou, and L. Jiang. 'Primer: Fast Private Transformer Inference on Encrypted Data'. In: *Proceedings of the 60th Annual ACM/IEEE Design Automation Conference*. 2025, pp. 1–6 (cited on page 211).