UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CHARLES FIGUEREDO DE BARROS

A PUBLIC-KEY ENCRYPTION SCHEME BASED ON LATTICE DEFORMATIONS

Rio de Janeiro
2016

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CHARLES FIGUEREDO DE BARROS

A PUBLIC-KEY ENCRYPTION SCHEME BASED ON LATTICE DEFORMATIONS

Tese de Doutorado submetida ao Corpo Docente do Programa de Pós-Graduação em Informática, do Instituto de Matemática e do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Doutor em Informática.

Orientador: Prof. Luis Menasché Schechter

Rio de Janeiro
2016

B277

Barros, Charles Figueredo de

A Public-Key Encryption Scheme Based on Lattice Deformations / Charles Figueredo de Barros. – 2016.

73 f.: il.

Orientador: Prof. Luis Menasché Schechter.

Tese (Doutorado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, 2016.

1. Criptografia de Chave Pública. 2. Reticulados. – I. Schechter, Prof. Luis Menasché (Orient.). II. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática. III. Título

CDD:

CHARLES FIGUEREDO DE BARROS

A PUBLIC-KEY ENCRYPTION SCHEME BASED ON LATTICE DEFORMATIONS

Tese de Doutorado submetida ao Corpo Docente do Programa de Pós-Graduação em Informática, do Instituto de Matemática e do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Doutor em Informática.

Aprovada em: Rio de Janeiro, 14 de DEZEMBRO de 2016.

Luis M. Schechter

Prof. Luis Menasché Schechter, D.Sc. (Orientador)

Vinicius Gusmão

Prof. Vinícius Gusmão Pereira de Sá, D.Sc.

Luziane F. de Mendonça

Profa. Luziane Ferreira de Mendonça, D.Sc.

S C. Coutinho

Prof. Severino Collier Coutinho, Ph.D.

P.B. Barreto

Prof. Paulo Sérgio Licciardi Messeder Barreto, D.Sc.

Rio de Janeiro
2016

Para meus pais.

AGRADECIMENTOS

When I look back	Quando olho para trás
I see the landscapes	Eu vejo as paisagens
That I have walked through	Pelas quais andei
But it is different	Mas elas estão diferentes
All the great trees are gone	Todas as grandes árvores se foram
It seems there are	Parece que há
Remnants of them	Restos delas
But it is the afterglow	Mas esta é a luz do ocaso
Inside of you	Dentro de você
Of all those you met	De todos aqueles que você conheceu
Who meant something in your life	E que significaram algo em sua vida

Olav Rex, August 1977

Aos meus pais, ao meu orientador, aos membros da banca de qualificação pelas contribuições valiosas que serviram para lapidar este trabalho, ao corpo docente do PPGI, aos amigos do LC3 e, por último mas não menos importante, à CAPES pelo suporte financeiro que foi essencial durante esses anos de estudo. Uma tese é apenas um brilho errante de conhecimento em meio à vasta escuridão da ignorância que nos cerca e desafia, mas ainda assim ilumina. Sozinho não se pode construir algo assim. Agradeço a todos que conheci (e que não conheci) e que significaram algo (certamente significaram e ainda significam muito) em minha vida.

RESUMO

BARROS, Charles Figueredo de. **A Public-Key Encryption Scheme Based on Lattice Deformations**. 2016. 73 f. Tese (Doutorado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016.

Neste trabalho, nós propomos a construção de um novo sistema de criptografia de chave pública, apresentado em duas versões. A primeira é baseada em operações com matrizes, enquanto a segunda é baseada em operações com polinômios, proporcionando chaves menores e operações mais rápidas de encriptação e decriptação. A segurança do sistema é baseada em um novo tipo de problema matemático da teoria de reticulados, que chamamos de problema da deformação de reticulados. A grosso modo, este problema consiste em, dada uma base deformada de um reticulado (em um sentido que será explicado em detalhes), encontrar a base original que sofreu a deformação.

Palavras-chave: Criptografia de Chave Pública. Reticulados.

ABSTRACT

BARROS, Charles Figueredo de. **A Public-Key Encryption Scheme Based on Lattice Deformations**. 2016. 73 f. Tese (Doutorado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2016.

We propose a new construction for a public-key cryptosystem, which is presented in two versions. The first is based on matrix operations, while the second is based on operations with polynomials, offering smaller keys, along with faster encryption and decryption. The security of the cryptosystem is based on a new kind of mathematical problem from the theory of lattices, which we call the Lattice Deformation Problem (LDP). Roughly speaking, the goal of this problem consists of, given a *deformed* lattice basis (in a sense that will be explained later in details), finding the original lattice basis which was deformed.

Keywords: Public-Key Cryptography. Lattices.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

"See the face of the shape of things to come." (Virtual Six)

Public-key cryptography has come a long way since the seminal work of Diffie and Hellman [1], whose research motivated Rivest, Shamir and Adleman to develop the first practical proposal of a public-key cryptosystem [2]. RSA, as it became known, changed the way secret information was transmitted over digital channels.

A public-key cryptosystem makes use of two different keys: one for encryption and another for decryption. The second is private, being held by the receiver of the secret communications; the first is derived from the private key and publicly distributed, being shared among those who send secret messages to the receiver. While the secret key is useless for encryption, the public key is useless for decryption. Therefore, sharing it does not compromise the secrecy of the communications.

What makes this possible is the existence of *trapdoor functions*. Roughly speaking, these are functions *easy* to compute but *hard* to invert without knowledge of some special information, called trapdoor. The information necessary to evaluate the function in its input domain can be derived from the trapdoor, but the reciprocal is not true.

Therefore, a public-key cryptosystem can be obtained from a trapdoor function by simply associating the trapdoor to the secret key, and the information needed to evaluate the function in its input domain to the public key. Encryption is equivalent to evaluating the function, while decryption consists of its inversion (with the aid of the trapdoor).

The security of a public-key cryptosystem is always based on some mathematical problem. There must be a relation between the keys, such that recovering the secret key from the public key should be equivalent to solving an instance of the problem (see Figure 1.1). As a consequence, this *underlying problem* must be hard to solve, otherwise it would be easy to retrieve the secret key from the public key, which is obviously not acceptable.

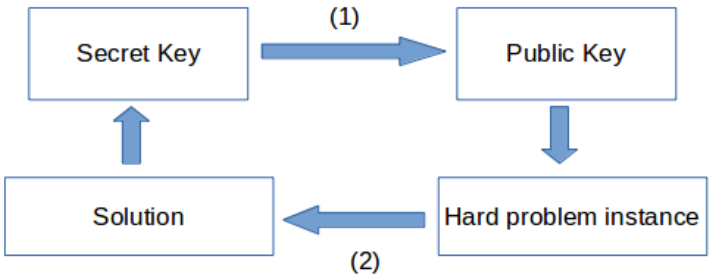

Figure 1.1: Deriving the public key from the secret key is easy (1), but in order to retrieve the secret key from the public key, one must solve an instance of a hard problem (2).

The precise notion of *hardness* will be presented in the next chapter, but for now it is sufficient to keep in mind that a problem is considered hard if no known algorithm can solve it in *reasonable time*. For example, the underlying problem of RSA is integer factorization. Taking into account that the best factoring algorithms would take longer than the age of the Universe to factor a 4096 bit RSA key, it becomes clear what we mean by *reasonable*.

In the mid '90s, Shor [3] demonstrated how to efficiently solve the problem of integer factorization (and discrete log computation). Fortunately, the task could only be accomplished by a quantum computer, which was nothing but a theoretical dream at that time. Consequently, the findings of Shor did not mean the death of public-key cryptosystems.

Nevertheless, this potentially dangerous dream triggered the development of a

whole new field of research known as *post-quantum cryptography*, which investigates classical cryptographic constructions (in the sense that they do not make use of any kind of quantum computing) that could resist, at least from a theoretical standpoint, to attacks by quantum (and classic) computers.

There is, in fact, a myriad of cryptographic constructions that are considered post-quantum, including those based on multivariate polynomials [4, 5, 6, 7], along with hash-based [8, 9, 10], code-based [11, 12, 13] and lattice-based cryptography [14, 15, 16, 17, 18, 19].

1.1 Lattice-Based Cryptography

Lattices have been used in a huge variety of cryptographic constructions, not limited to traditional public-key encryption schemes, but also digital signature schemes, cryptographic hash functions, homomorphic encryption schemes and authentication protocols.

It is safe to say that two main problems, or its variants, underlie all lattice-based public-key cryptosystems: the *shortest vector problem* (SVP) and the *closest vector problem* (CVP). This is true even for the modern constructions based on LWE (Learning With Errors), since it can be shown that this problem is also related to CVP.

This work goes in a slightly different direction by proposing a new kind of problem which, at first sight, is not related to SVP or CVP at all. Nevertheless, we establish a connection between our problem and that of finding relatively short vectors in lattices, but not necessarily the shortest one.

1.2 Thesis Roadmap

This thesis is structured as follows: in Chapter 2, the theoretical background is provided. We present the basics on linear algebra, lattices and other topics that will be necessary for the complete understanding of this work. Chapter 3 is where we begin our contribution by introducing a new trapdoor function. In Chapter 4, a cryptosystem is built upon the trapdoor function which was presented in the previous chapter. In Chapter 5, we assess the hardness of exposing the secret key of the proposed cryptosystem, introducing a new mathematical problem from the theory of lattices. A variant of the proposed cryptosystem is presented in Chapter 6. Finally, in Chapter 7, we present our final remarks and conclusions.

2 THEORETICAL BACKGROUND

"Okay... we're gonna have to do this the hard way then." (Starbuck)

In this chapter, we review the basics on linear algebra, complexity theory and lattices. We also establish notations and conventions that will be adopted in the rest of this thesis.

2.1 Linear Algebra

Throughout this work, vectors will be considered as *row vectors* and represented by lowercase boldface letters, such as $\mathbf{u}$. The entries will be represented by $\mathbf{u}_k$, with k starting from 0. We also employ the symbol $\lceil \mathbf{u} \rfloor$ to denote the vector in which each entry is the nearest integer to the corresponding entry of $\mathbf{u}$. We are going to use the special notation $\mathbf{e}^{(k)}$ to represent the vector whose entry at position k is 1, and all the other entries are 0. Concatenation of two vectors $\mathbf{u}$ and $\mathbf{v}$ will be denoted by $\mathbf{u}||\mathbf{v}$.

Matrices will be represented by uppercase letters, such as A. Given a square matrix A of dimension n, the element on row i and column j of A will be denoted by $A_{i,j}$, for $i, j \in \{0, 1, \cdots, n-1\}$. For any $0 \leq k \leq n-1$, row k of A will be represented by A_k. The notation $A > 0$ (conversely $A < 0$) implies that all the entries of A are greater (or less) than 0. Finally, the determinant of A will be denoted by $\det(A)$. In this work, we are interested in the following special classes of matrices:

Definition 2.1 *An integer square matrix U is said to be unimodular if* $|\det(U)| = 1$.

Definition 2.2 *A real square matrix S is said to be an M-matrix if it can be written as $S = \gamma I + Q$, where $Q \leq 0$ and $\gamma \geq \rho(Q)$.*[1]

A useful property of such M-matrices is known as *inverse-positivity*. We state this property in the form of a theorem. For the proof, [20] can be consulted.

Theorem 2.1 *If $S = \gamma I + Q$ is an M-matrix with $\gamma > \rho(Q)$, then S^{-1} exists and $S^{-1} \geq 0$ (S is an inverse-positive matrix).*

The following definition is nonstandard, but it will be helpful, since it summarizes information on how an M-matrix is built.

Definition 2.3 *Given two positive integers γ and τ, a real matrix S is said to be an $M_{\gamma,\tau}$-matrix if it is a nonsingular M-matrix and can be written as $S = \gamma I + Q$, with $Q_{i,j} \in \{-\tau, -\tau + 1, \cdots, -1, 0\}$.*

2.2 Complexity Theory

When dealing with algorithms, we are mostly interested in knowing how efficient they are. In an ultimate analysis, we investigate the hardness of the problems we are trying to solve by employing those algorithms. If, after extensive research, no one could find a good algorithm to solve a specific problem, or the algorithms

[1] Here, $\rho(Q)$ denotes the spectral radius of Q, i.e., $\rho(Q) = \max\{|\xi_1|, \cdots, |\xi_n|\}$, where $\xi_1, \cdots, \xi_n$ are the eigenvalues of Q.

which were found are not efficient, it is reasonable to conjecture that this problem is *computationally hard*.

The main goal of Complexity Theory is to establish precise definitions of what is *hard* and what is *easy*, what is *efficient* and what is *inefficient*.

Complexity theory investigates how much resources, such as time and memory space, are required to solve certain computational problems. Although it seems to be common sense that computers are powerful enough to solve any kind of mathematical problem, as we will see, this is far from being true. While it is true that some problems are *easy* to solve, there are a bunch of problems which apparently cannot be *efficiently* solved even by the most powerful computers in the world.

In order to analyse the complexity of problems, we must precisely define what *hard* and *easy* mean from a computational standpoint. Typically, we measure the complexity of an algorithm in terms of its input size, and this measure is expressed as the running time of the algorithm. There are two ways of doing so:

1. Empirically, by actually running the algorithm and computing the time it takes to run over a certain input.
2. Theoretically, by obtaining a mathematical expression of the algorithm's running time as a function of its input size.

The second approach has the advantage of not depending on specific conditions, such as processing resources, compilers and programming languages. However, it is a much harder approach. In order to make it easier, we employ some simplifications, such as discarding additive and multiplicative constants and considering only the

asymptotic behaviour of the algorithms (when the input size is large enough).

2.2.1 Asymptotic Notation

Asymptotic notations capture the simplifications adopted throughout theoretical complexity analysis. They serve as a powerful tool to measure the efficiency of algorithms. From now on, let f and g be non-negative real functions of a natural variable n.

Definition 2.4 *We say that $f = O(g)$ if there is a positive integer n_0 and a positive real constant c such that*

$$f(n) \leq cg(n) \tag{2.1}$$

for all $n > n_0$.

In other words, f is asymptotically limited by g, which means that, for a sufficiently large n, the growth rate of f does not exceed the growth rate of g except by a constant factor. In an entirely analogous way, we define the following asymptotic notations:

Definition 2.5 *We say that $f = \Omega(g)$ if there is a positive integer n_0 and a positive real constant c such that*

$$f(n) \geq cg(n) \tag{2.2}$$

for all $n > n_0$.

Definition 2.6 *We say that $f = \theta(g)$ if $f = O(g)$ and $g = O(f)$.*

While the O-notation is used for upper bounds, the Ω-notation applies to lower bounds, and the θ-notation is adopted to express tight bounds. As we have already mentioned, these notations capture simplifications such as discarding constants and considering only asymptotic behavior. Hence, we can say that $n^3 + 3n^2 = O(n^3)$, as well as $3000n^3 - 230n = O(n^3)$. Similarly, $2^n + 3000n^{10} = O(2^n)$ and $30\log\ n + 20000 = O(\log\ n)$. Given a function f, the following asymptotic bounds are the most widely used:

1. Constant: $f = O(1)$.
2. Logarithmic: $f = O(\log\ \mathrm{n})$.
3. Linear: $f = O(n)$.
4. Quadratic: $f = O(n^2)$.
5. Cubic: $f = O(n^3)$.
6. Exponential: $f = O(2^n)$.

More generally, we may refer to n^c, for $c > 0$, as a *polynomial bound*. Another commonly used asymptotic bound is $O(n\log\ n)$, which represents an intermediate between linear and quadratic complexities. Figure 2.1 shows a comparative analysis of these growth rates.

Algorithms with constant, logarithmic or polynomial complexity are considered *efficient*, while exponentially bounded algorithms are said to be inefficient. Conversely, the problems for which logarithmic or polynomial algorithms are known can be considered *easy* (efficiently solvable), but problems for which the best known algorithms are exponential can be considered *hard*.

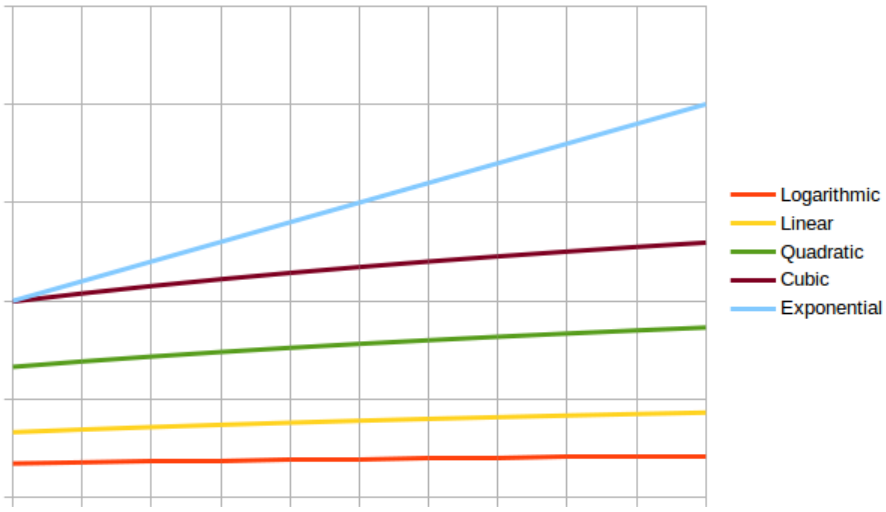

Figure 2.1: Asymptotic bounds comparative.

It is important to keep in mind that this analysis does not take into account the specific nature of the input, implying in a series of simplifications. Even an instance of a hard problem can be easy to solve. For example, it is easy to factor a 10 bit integer. Conversely, a polynomial-time algorithm may take really long to run over a large input, maybe due to the large degree of the polynomial, or even discarded multiplicative constants that can affect the running time of the algorithm in practical scenarios. Nevertheless, the asymptotic analysis is an extremely helpful tool to measure algorithmic complexity and theoretical hardness.

2.2.2 Complexity Classes

Computational problems can be categorized according to their theoretical hardness. A complexity class is a set of problems whose complexities are somehow related. This categorization is based on some kind of resource, typically time and space. The complexity class reflects how the requirement in resources grow as the input size increases.

Complexity classes are typically related to *decision problems*. We may think of a decision problem as a question which admits one of two answers: yes or no. For example, given two graphs, decide if they are isomorphic. The class P contains all decision problems that can be solved in polynomial time. These are the problems considered tractable or efficiently solvable. On the other hand, the class NP contains all decision problems for which the "yes" instances can be efficiently verified. In other words, given a "yes" instance, we can verify in polynomial time that it is, in fact, a "yes" instance.

In order to compare the hardness of different problems, we use the concept of *reduction*. Roughly speaking, a reduction consists of transforming one problem into another. This method is widely used to prove that a certain problem is at least as hard (or as easy) as another. If the transformation can be made in polynomial time, we say that it is a *polynomial reduction*.

Put into more formal terms, let A and B be two decision problems, with input domains $\mathcal{A}$ and $\mathcal{B}$ respectively. A polynomial reduction from A to B is a polynomial-time function $\sigma : \mathcal{A} \longrightarrow \mathcal{B}$ such that $a \in \mathcal{A}$ is a "yes" instance if $\sigma(a) \in \mathcal{B}$ is a "yes" instance.

We denote a polynomial-time reduction from A to B by $A \propto B$. Reducing A to B allows us to solve A by solving B. As already mentioned, reductions are useful tools to prove that a problem is as hard/easy as another.

Assume that A is a hard problem. In order to prove that B is at least as hard as A, we reduce A to B and show that, if we can solve B, we can solve A. On the other hand, if A is an *easy problem* (for which a polynomial-time algorithm is known), we can solve B by reducing B to A.

A decision problem A is said to be NP-complete if $A \in NP$ and every problem of NP reduces to A. If the latter is true, but we cannot prove that $A \in NP$, then A is said to be NP-hard. NP-complete problems are considered the hardest problems of the class NP, because an efficient solution for any of them would imply a solution for all NP problems. In fact, a polynomial-time solution for one single NP-complete problem would give a definitive answer for the long standing open question as to whether P = NP, and it would be "yes".

2.3 Lattices

Lattices play an essential role in the design of various post-quantum cryptographic constructions. Algebraically speaking, a lattice is a discrete additive subgroup of $\mathbb{R}^n$.

Definition 2.7 *Let n be a positive integer. Given a real $n \times n$ matrix A, the lattice generated by A is the set*

$$\mathcal{L}(A) = \{\mathbf{u}A : \ \mathbf{u} \in \mathbb{Z}^n\}. \tag{2.3}$$

In other words, the set $\mathcal{L}(A)$ corresponds to all the integer linear combinations of the rows of A. In particular, if A is an integer matrix, we say that $\mathcal{L}(A)$ is an integral lattice. The matrix A is said to be a *basis* of the lattice. Like a vector space, a lattice admits infinitely many bases (see Figure 2.2).

Definition 2.8 *Given two real matrices A, B, we say that $\mathcal{L}(A)$ is a sublattice of $\mathcal{L}(B)$ if there exists an integer matrix R such that $A = RB$.*

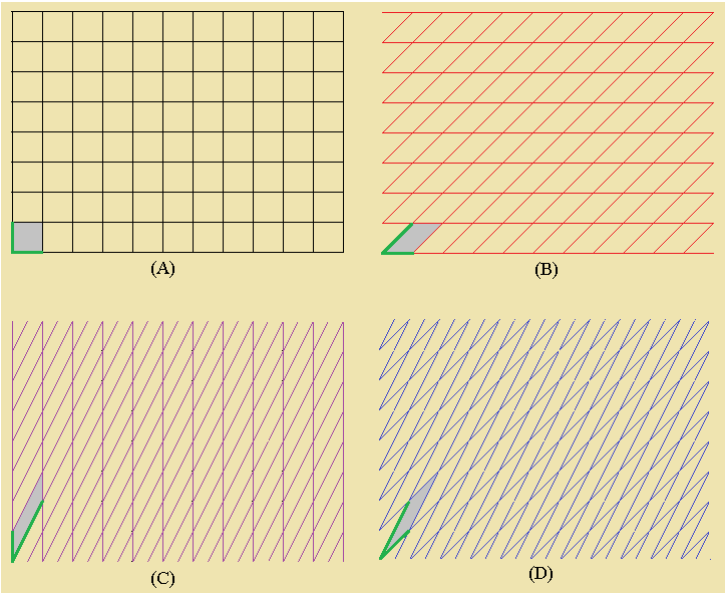

Figure 2.2: The same lattice from the standpoint of four different bases.

In particular, if R is unimodular, it can be proven that $\mathcal{L}(A) = \mathcal{L}(B)$. As a matter of fact, $\mathcal{L}(A) = \mathcal{L}(B)$ iff there is a unimodular matrix U such that $A = UB$.

Theorem 2.2 *Let A and B be two $n \times n$ real matrices. Then, $\mathcal{L}(A) = \mathcal{L}(B)$ iff there is a unimodular matrix U such that $A = UB$.*

Proof: Assume that there is a unimodular matrix U such that $A = UB$, and let $\mathbf{u} \in \mathcal{L}(A)$. Hence, $\mathbf{u} = \mathbf{w}A$, for some $\mathbf{w} \in \mathbb{Z}^n$ and, consequently, $\mathbf{u} = \mathbf{w}UB$. Since $\mathbf{w}U \in \mathbb{Z}^n$, we have that $\mathbf{u} \in \mathcal{L}(B)$. On the other hand, consider $\mathbf{u} \in \mathcal{L}(B)$. Hence, $\mathbf{u} = \mathbf{w}B$, for some $\mathbf{w} \in \mathbb{Z}^n$ and, consequently, $\mathbf{u} = \mathbf{w}U^{-1}A$. Since U^{-1} is integer, we have that $\mathbf{w}U^{-1} \in \mathbb{Z}^n$ and, therefore, $\mathbf{u} \in \mathcal{L}(A)$.

Now assume that $\mathcal{L}(A) = \mathcal{L}(B)$. Hence, every vector of A can be written as a linear combination with integer coefficients of the vectors from B, which gives us an integer coefficient matrix U such that $A = UB$. Reciprocally, every vector of B

can be written as a linear combination with integer coefficients of the vectors from A, which gives us the relation $B = U^{-1}A$. Since U^{-1} must be an integer matrix and $\det(U^{-1}) = 1/\det(U)$, we conclude that $\det(U) = \pm 1$. $\square$

As already mentioned, the mainstream of lattice-based cryptographic constructions is based upon two problems: the *shortest vector problem* (SVP) and the *closest vector problem* (CVP). The first one asks for the shortest nonzero vector of a given lattice, while the second consists of, given a lattice and an arbitrary point, finding the lattice vector which is closest to that point. Both problems are considered hard. As a matter of fact, SVP is known to be NP-hard for randomized reductions [21].

It is worth mentioning that whenever we say that we are given a lattice, we mean a lattice basis. Although solving SVP exactly is a hard task, some techniques allow us to obtain an approximation of the shortest vector. These techniques are known as *lattice reduction* methods, but their approximation power is not unlimited. In [22], it was demonstrated that approximating the shortest vector within a factor less than $\sqrt{2}$ is also NP-hard for randomized reductions.

CVP can be solved by the rounding technique, provided that we know a *good* basis for the lattice. A basis is considered good if its vectors are short and form large angles (close to 90 degrees) among each other. Consequently, a *bad* basis has large vectors with small angles among each other (see Figure 2.3).

Any CVP instance is a vector of the form

$$\mathbf{c} = \mathbf{u} + \mathbf{r} \tag{2.4}$$

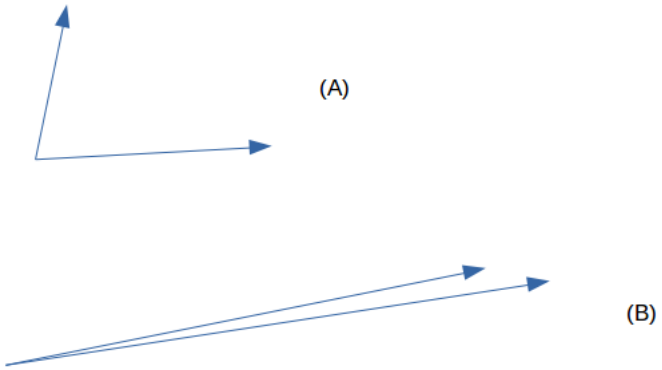

Figure 2.3: A good basis (A) and a bad basis (B).

where $\mathbf{u}$ is the lattice point we would like to find. Given a good lattice basis B, we may write $\mathbf{u} = \mathbf{x}B$ for some $\mathbf{x} \in \mathbb{Z}^n$. Hence,

$$\mathbf{c}B^{-1} = \mathbf{x} + \mathbf{r}B^{-1}. \tag{2.5}$$

Rounding to the nearest integer, we obtain

$$\lceil \mathbf{c}B^{-1} \rfloor = \mathbf{x} + \lceil \mathbf{r}B^{-1} \rfloor. \tag{2.6}$$

If $\mathbf{r}$ is a sufficiently short vector (if $\mathbf{u}$ is the lattice vector closest to $\mathbf{c}$, then $\mathbf{r}$ *must* be sufficiently short) and B is a good basis, then $\lceil \mathbf{r}B^{-1} \rfloor = \mathbf{0}$. In other words, rounding $\mathbf{c}B^{-1}$ gives us the coefficient vector of $\mathbf{u}$ in the basis B and, consequently, the vector $\mathbf{u}$ itself.

This technique is due to Babai [23] and is at the core of the GGH [24] decryption procedure.

3 BUILDING A NEW TRAPDOOR FUNCTION

"Out of the box is where I live." (Starbuck)

We begin this chapter by reviewing the basic concept of trapdoor functions, which play an essential role in the design of public-key cryptosystems. We also introduce the nonstandard concepts of evaluation set and trapdoor set for a given trapdoor function.

Definition 3.1 *A function f is a trapdoor function if it can be efficiently evaluated on its input domain, but its inversion requires the knowledge of some special information, called the trapdoor.*

Definition 3.2 *An evaluation set for a trapdoor function f is a set $\mathcal{E}$ of public parameters, required for the evaluation of f on its input domain. In this case, we use the notation $f_{\mathcal{E}}$ to represent the function f with evaluation set $\mathcal{E}$.*

Definition 3.3 *A trapdoor set for the inverse of a trapdoor function f is a set $\mathcal{T}$ of secret parameters, required for the evaluation of f^{-1} on its input domain. In this case, we use the notation $f_{\mathcal{T}}^{-1}$ to represent the inverse function f^{-1} with trapdoor set $\mathcal{T}$.*

3.1 Construction and Evaluation

From now on, for any positive integers a, b such that $a \leq b$, we define the sets $\mathcal{Z}_a = \{-a, -a+1, \cdots, -1, 0\}$, $\mathcal{Z}^b = \{0, 1, \cdots, b\}$ and $\mathcal{Z}_a^b = \mathcal{Z}_a \cup \mathcal{Z}^b$. We describe below the construction of our trapdoor function:

- Select a positive integer n, which is the security parameter.
- Pick up positive integers $\alpha, \beta, \gamma, \delta, \tau, \sigma, \omega$ and positive real numbers ϵ_1, ϵ_2. A heuristic choice for these parameters will be discussed later.
- Build an integer $n \times n$ matrix E, with $E_{i,j}$ randomly and uniformly selected from $\mathcal{Z}_\alpha^\beta$, for all $0 \leq i, j < n$.
- Build a random $n \times n$ unimodular matrix $U = RT'TC$, where T' is lower triangular, T is upper triangular, R and C are random permutation matrices. The diagonal entries of T and T' must be randomly and uniformly chosen from the set $\{-1, 1\}$, while the other entries should be selected uniformly at random from the set $\mathcal{Z}_\omega^\omega$.
- Build an $n \times n$ $M_{\gamma,\tau}$-matrix $S = \gamma I + Q$, with $Q_{i,j}$ selected uniformly at random from the set $\mathcal{Z}_\tau$. The inverse of S must satisfy the following conditions:

$$\frac{1}{\gamma} < S_{j,j}^{-1} < \frac{1+\epsilon_1}{\gamma}, \text{ for all } j = 0, \cdots, n-1 \tag{3.1}$$

and

$$0 < S_{i,j}^{-1} < \frac{\tau(1+\epsilon_2)}{\gamma^2}, \text{ for all } i \neq j. \tag{3.2}$$

- Build the matrix $P = US + E$.
- Choose integer numbers $\theta_1, \theta_2, \mu_1, \mu_2$ satisfying the following conditions:

$$\theta_2 > \theta_1 > n\sigma\alpha > 0, \tag{3.3}$$

$$\mu_1 < \mu_2 < -n\sigma\beta < 0, \tag{3.4}$$

$$\theta_1 > \gamma\delta + n\sigma\alpha - \frac{\tau(n-1)(\mu_1 - n\sigma\alpha)(1+\epsilon_2)}{\gamma}, \tag{3.5}$$

$$\theta_2 < \frac{\gamma^2(2\delta+1)}{2(\gamma(1+\epsilon_1)+\tau(n-1)(1+\epsilon_2))} - n\sigma\beta, \tag{3.6}$$

$$\mu_1 > \frac{-\gamma^2(2\delta+1)}{2(\gamma(1+\epsilon_1)+\tau(n-1)(1+\epsilon_2))} + n\sigma\alpha \tag{3.7}$$

and

$$\mu_2 < -\gamma\delta - n\sigma\beta - \frac{\tau(n-1)(\theta_2 + n\sigma\beta)(1+\epsilon_2)}{\gamma}. \tag{3.8}$$

A reasonable justification for all these conditions will be presented shortly. For now, it is sufficient to say that they ensure that the inversion algorithm for our trapdoor function works properly.

We fix $\mathcal{E} = \{P, n, \sigma, \theta_1, \theta_2, \mu_1, \mu_2\}$ as the evaluation set of our trapdoor function. Define the non-deterministic mapping

$$G : \mathbb{N} \times \mathbb{Z}^4 \longrightarrow \mathbb{Z}^n \tag{3.9}$$

which, on input $n, \theta_1, \theta_2, \mu_1, \mu_2$, outputs a vector $\mathbf{r} \in \mathbb{Z}^n$ as follows:

- For all $k = 0, 1, \cdots, n-1$, it generates a random bit b_k;
- If $b_k = 0$, then $\mathbf{r}_k$ is a random integer uniformly chosen from the interval $[\theta_1, \theta_2]$;
- If $b_k = 1$, then $\mathbf{r}_k$ is a random integer uniformly chosen from the interval $[\mu_1, \mu_2]$.

Our function takes as input a vector $\mathbf{x} \in \mathbb{Z}^n$, with entries from the set $\mathcal{Z}^\sigma$, and outputs

$$f_{\mathcal{E}}(\mathbf{x}) = \mathbf{r} + \mathbf{x}P, \tag{3.10}$$

where $\mathbf{r} = G(n, \theta_1, \theta_2, \mu_1, \mu_2)$.

3.2 Inversion

The next step consists of establishing a proper inversion algorithm, making use of the trapdoor set $\mathcal{T} = \{U^{-1}, S^{-1}, \delta\}$. From now on, let $\mathbf{c} = f_{\mathcal{E}}(\mathbf{x}) = \mathbf{r} + \mathbf{x}P$. Define the *message residue vector*

$$\mathbf{v} = \mathbf{r} + \mathbf{x}E, \tag{3.11}$$

which represents, roughly speaking, the portion of $\mathbf{c}$ which we would like to get rid of in order to retrieve $\mathbf{x}$. Note that $\mathbf{c} = \mathbf{u} + \mathbf{v}$, where $\mathbf{u} = \mathbf{x}US \in \mathcal{L}(S)$. Our goal is to figure out a way of retrieving the coefficient vector (relatively to the basis S) of the lattice point $\mathbf{u}$, namely $\mathbf{x}U$. Define the *inversion error vector*

$$\mathbf{e} = (\mathbf{r} + \mathbf{x}E)S^{-1} = \mathbf{v}S^{-1}. \tag{3.12}$$

Provided that $\mathbf{c}$ is a perturbation over a vector of $\mathcal{L}(S)$, the most natural decoding method consists of rounding to the nearest lattice point, using the technique discussed at the end of Chapter 2. However, since the perturbation $\mathbf{v}$ is not small, the rounding does not lead us back to the vector $\mathbf{u}$, and $\lceil \mathbf{e} \rfloor$ represents this rounding error. As will be demonstrated shortly, each entry of this error vector lies in a quite narrow interval.

Lemma 3.1 *For all $j = 0, 1, \cdots, n-1$, the following holds:*

$$\mu_1 - n\sigma\alpha < \mathbf{v}_j < \theta_2 + n\sigma\beta. \tag{3.13}$$

Proof: Note that

$$\mathbf{v}_j = \mathbf{r}_j + \sum_{i=0}^{n-1} \mathbf{x}_i E_{i,j}, \tag{3.14}$$

for all $j = 0, 1, \cdots, n-1$. The inequalities in (3.13) follow immediately from the fact that $\mu_1 < \mathbf{r}_j < \theta_2$, $0 \leq \mathbf{x}_j \leq \sigma$ and $-\alpha \leq E_{i,j} \leq \beta$. $\square$

Lemma 3.2 *Let* $0 \leq j \leq n-1$. *If* $\theta_1 < \mathbf{r}_j < \theta_2$, *then*

$$\theta_1 - n\sigma\alpha < \mathbf{v}_j < \theta_2 + n\sigma\beta. \tag{3.15}$$

On the other hand, if $\mu_1 < \mathbf{r}_j < \mu_2$, *then*

$$\mu_1 - n\sigma\alpha < \mathbf{v}_j < \mu_2 + n\sigma\beta. \tag{3.16}$$

Proof: The proof is similar to the previous lemma, taking into account the new bounds for $\mathbf{r}_j$. □

The following two lemmas show that the inversion error vector is controlled by the vector $\mathbf{r}$, and its entries lie in a quite narrow interval.

Lemma 3.3 *Let* $0 \leq j \leq n-1$. *If* $\theta_1 < \mathbf{r}_j < \theta_2$ *and conditions* (3.1) *to* (3.8) *hold, then*

$$\delta < \mathbf{e}_j < \delta + \frac{1}{2}. \tag{3.17}$$

Proof: From the definition of the inversion error vector in (3.12), we have

$$\mathbf{e}_j = \mathbf{v}_j S^{-1}_{j,j} + \sum_{i \neq j} \mathbf{v}_i S^{-1}_{i,j}, \tag{3.18}$$

which yields

$$\mathbf{e}_j \geq \mathbf{v}_j S^{-1}_{j,j} + (n-1) \min_{i \neq j} \{\mathbf{v}_i\} \max_{i \neq j} \left\{S^{-1}_{i,j}\right\}. \tag{3.19}$$

Note that we combined the lower bound for $\mathbf{v}_i$ with the upper bound of $S^{-1}_{i,j}$, because $S^{-1}_{i,j} > 0$ for all $0 \leq i, j \leq n-1$ and the lower bound for $\mathbf{v}_i$, when $i \neq j$, is a negative number by Lemma 1. Hence, using (3.1), (3.2), (3.13) and (3.15), we obtain

$$\mathbf{e}_j > \frac{\theta_1 - n\sigma\alpha}{\gamma} + \frac{\tau(n-1)(\mu_1 - n\sigma\alpha)(1+\epsilon_2)}{\gamma^2}. \tag{3.20}$$

Applying condition (3.5) above yields exactly $\mathbf{e}_j > \delta$. On the other hand, by (3.18) we have that

$$\mathbf{e}_j \leq \mathbf{v}_j S^{-1}_{j,j} + (n-1)\max_{i \neq j}\{\mathbf{v}_i\} \max_{i \neq j}\left\{S^{-1}_{i,j}\right\}. \tag{3.21}$$

By (3.1), (3.2), (3.13) and (3.15), we get

$$\mathbf{e}_j < (\theta_2 + n\sigma\beta)\left(\frac{1+\epsilon_1}{\gamma} + \frac{\tau(n-1)(1+\epsilon_2)}{\gamma^2}\right). \tag{3.22}$$

Using (3.6), we obtain $\mathbf{e}_j < \delta + 1/2$. □

Lemma 3.4 *Let* $0 \leq j \leq n-1$. *If* $\mu_1 < \mathbf{r}_j < \mu_2$ *and conditions* (3.1) *to* (3.8) *hold, then*

$$-\delta - \frac{1}{2} < \mathbf{e}_j < -\delta. \tag{3.23}$$

Proof: We keep (3.21) in mind. Using (3.1), (3.2), (3.13) and (3.16), we obtain

$$\mathbf{e}_j < \frac{\mu_2 + n\sigma\beta}{\gamma} + \frac{\tau(n-1)(\theta_2 + n\sigma\beta)(1+\epsilon_2)}{\gamma^2}. \tag{3.24}$$

Using (3.8), we get $\mathbf{e}_j < -\delta$. Finally, using the lower bound for $\mathbf{e}_j$ given by (3.19) and taking into account that $\mathbf{v}_j < 0$, we obtain

$$\mathbf{e}_j > (\mu_1 - n\sigma\alpha)\left(\frac{1+\epsilon_1}{\gamma} + \frac{\tau(n-1)(1+\epsilon_2)}{\gamma^2}\right). \tag{3.25}$$

Combined with condition (3.7), the inequality above yields $\mathbf{e}_j > -\delta - 1/2$. □

Lemmas 3 and 4 provide us with crucial information: the entries of $\mathbf{e}$ always round to either δ or $-\delta$. Unfortunately, it is determined by $\mathbf{r}$, which is not known. Initially, it appears to be a drawback on our intentions to use vector $\mathbf{e}$ to build an inverse to our trapdoor function. However, as will be shown in the next lemma, the rounded entries of $\mathbf{e}$ can still be determined, even with no prior knowledge of $\mathbf{r}$. Recall that $\mathbf{c} = f_{\mathcal{E}}(\mathbf{x}) = \mathbf{r} + \mathbf{x}P$. Define the *rounding difference vector* as follows:

$$\mathbf{d} = \mathbf{c}S^{-1} - \lceil \mathbf{c}S^{-1} \rfloor. \tag{3.26}$$

Essentially, this vector allows us to determine which entries of $\mathbf{e}$ round to δ and which entries round to $-\delta$.

Lemma 3.5 *For any $0 \leq j \leq n-1$, the following hold:*

1. $\mathbf{d}_j < 0$ *iff* $\lceil \mathbf{e}_j \rfloor = -\delta$;
2. $\mathbf{d}_j > 0$ *iff* $\lceil \mathbf{e}_j \rfloor = \delta$.

Proof: Indeed,

$$\mathbf{d} = (\mathbf{r} + \mathbf{x}P)S^{-1} - \lceil (\mathbf{r} + \mathbf{x}P)S^{-1} \rfloor. \tag{3.27}$$

Replacing P by $US + E$, we obtain

$$\mathbf{d} = (\mathbf{r} + \mathbf{x}E)S^{-1} + \mathbf{x}U - \lceil (\mathbf{r} + \mathbf{x}E)S^{-1} + \mathbf{x}U \rfloor. \tag{3.28}$$

Since $\mathbf{x}U$ is an integer vector, we have that

$$\lceil (\mathbf{r} + \mathbf{x}E)S^{-1} + \mathbf{x}U \rfloor = \lceil (\mathbf{r} + \mathbf{x}E)S^{-1} \rfloor + \mathbf{x}U. \tag{3.29}$$

Hence,

$$\mathbf{d} = (\mathbf{r} + \mathbf{x}E)S^{-1} - \lceil (\mathbf{r} + \mathbf{x}E)S^{-1} \rfloor = \mathbf{e} - \lceil \mathbf{e} \rfloor. \tag{3.30}$$

We already know that either $\delta < \mathbf{e}_j < \delta + 1/2$ or $-\delta - 1/2 < \mathbf{e}_j < -\delta$, for all $j = 0, \cdots, n-1$. If $\mathbf{d}_j > 0$, then $\mathbf{e}_j > \lceil \mathbf{e}_j \rfloor$, which only is possible when $\delta < \mathbf{e}_j < \delta + 1/2$. On the other hand, if $\mathbf{d}_j < 0$, we have that $\mathbf{e}_j < \lceil \mathbf{e}_j \rfloor$, which only occurs when $-\delta - 1/2 < \mathbf{e}_j < -\delta$. The reciprocal of these implications is straightforward. Hence, we conclude that $\mathbf{d}_j > 0$ if and only if $\lceil \mathbf{e}_j \rfloor = \delta$. Similarly, $\mathbf{d}_j < 0$ iff $\lceil \mathbf{e}_j \rfloor = -\delta$. □

Provided with the previous results, we can build an inversion algorithm for our function, using the trapdoor set $\mathcal{T} = \{U^{-1}, S^{-1}, \delta\}$, as follows:

1. On input $\mathbf{c} \in \mathbb{Z}^n$, compute $\mathbf{d} = \mathbf{c}S^{-1} - \lceil \mathbf{c}S^{-1} \rfloor$;
2. Initialize a vector $\mathbf{e}' \in \mathbb{Z}^n$ with zeros. For $j = 0, 1, \cdots, n-1$:
   (a) If $\mathbf{d}_j > 0$, set $\mathbf{e}'_j = \delta$;
   (b) Otherwise, set $\mathbf{e}'_j = -\delta$;
3. Output the vector

$$\mathbf{y} = (\lceil \mathbf{c}S^{-1} \rfloor - \mathbf{e}')U^{-1}. \tag{3.31}$$

In the next proposition, we show that the procedure above actually inverts the function $f_{\mathcal{E}}$. In other words, we prove that $\mathbf{y} = f_{\mathcal{T}}^{-1}(f_{\mathcal{E}}(\mathbf{x})) = \mathbf{x}$, for all $\mathbf{x}$ on the input domain of $f_{\mathcal{E}}$.

Proposition 3.1 *Let* $\mathbf{c} = f_{\mathcal{E}}(\mathbf{x})$*, with* $\mathbf{x}$ *on the input domain of* $f_{\mathcal{E}}$*, and* $\mathbf{y}$ *as defined in the procedure above. The following holds:*

$$\mathbf{y} = f_{\mathcal{T}}^{-1}(f_{\mathcal{E}}(\mathbf{x})) = \mathbf{x}. \tag{3.32}$$

Proof: Replacing $\mathbf{c}$ by $\mathbf{r} + \mathbf{x}P$, we obtain

$$\mathbf{y} = (\lceil (\mathbf{r} + \mathbf{x}P)S^{-1} \rfloor - \mathbf{e}')U^{-1}, \tag{3.33}$$

which yields, after replacing P by $US + E$,

$$\mathbf{y} = (\lceil (\mathbf{r} + \mathbf{x}E)S^{-1} + \mathbf{x}U \rfloor - \mathbf{e}')U^{-1}. \tag{3.34}$$

Using the fact that $\mathbf{x}U$ is an integer vector, we may write the previous identity as follows:

$$\mathbf{y} = (\lceil (\mathbf{r} + \mathbf{x}E)S^{-1} \rfloor + \mathbf{x}U - \mathbf{e}')U^{-1}. \tag{3.35}$$

By construction, we have that $\mathbf{e}' = \lceil \mathbf{e} \rfloor$. Therefore, we finally obtain

$$\mathbf{y} = (\lceil (\mathbf{r} + \mathbf{x}E)S^{-1} \rfloor + \mathbf{x}U - \lceil (\mathbf{r} + \mathbf{x}E)S^{-1} \rfloor)U^{-1} = \mathbf{x}, \tag{3.36}$$

for all $\mathbf{x}$ in the domain of $f_{\mathcal{E}}$. Therefore, $\mathbf{y} = f_{\mathcal{T}}^{-1}(f_{\mathcal{E}}(\mathbf{x}))$. $\square$

It is worth mentioning that, even though there is a random component in our trapdoor function (the vector $\mathbf{r}$), the inversion algorithm is deterministic. In other words, the same input $\mathbf{x}$ can be mapped to different values, but all these values are mapped back to the same vector $\mathbf{x}$ by the inversion algorithm.

Some questions remain open and will be answered later on this thesis. One of the most important is whether it is possible to choose parameters satisfying conditions (3.1) to (3.8). In this section, we simply required these conditions to hold, but no guarantees were given as for the existence of parameters fulfilling these requirements. In the next chapter, we present possible choices of parameters satisfying the aforementioned conditions.

4 FROM A TRAPDOOR FUNCTION TO A CRYPTOSYSTEM

> "So the fate of the entire human race depends upon my wild guess."
> (Dr. Gaius Baltar)

We build an encryption scheme from our trapdoor function in the canonical way, using the evaluation set as public key and the trapdoor set as the secret key. A full cryptosystem from our function requires procedures for key generation, encryption and decryption. Key generation involves the following algorithms:

- `PARAMETERS`: takes as input the values of n, α, β, γ, δ, τ, σ, ω, ϵ_1, ϵ_2 and outputs the parameters $\theta_1, \theta_2, \mu_1, \mu_2$, which must satisfy conditions (3.3) to (3.8).
- `M_MATRIX`: on input $n, \gamma, \tau, \epsilon_1, \epsilon_2$, outputs an $M_{\gamma,\tau}$-matrix S of dimension n, satisfying conditions (3.1) and (3.2).
- `UNIMODULAR`: on input n, ω, outputs a random n-dimensional unimodular matrix U, as described in the previous chapter.
- `RANDOM_MATRIX`: on input n, α, β, outputs an n-dimensional matrix E whose entries are uniformly chosen from the set $\mathcal{Z}_\alpha^\beta$.

An instance of a secret key is given by the set $\texttt{SK} = \{S^{-1}, U^{-1}, \delta\}$, while the corresponding public key is the set $\texttt{PK} = \{P, n, \sigma, \theta_1, \theta_2, \mu_1, \mu_2\}$, where $P = US + E$.

Encryption consists of computing the trapdoor function of the previous chapter on the input $\mathbf{x}$, where the vector $\mathbf{x}$ must encode the plaintext. The following

procedures are required:

- `ENCODE`: takes as input a binary stream b and outputs $\mathbf{x} \in \mathbb{Z}^n$, with entries in $\mathcal{Z}^\sigma$, which encodes b. An example of an encoding procedure is shown later in this chapter.
- `EPHEMERAL_KEY`: outputs $\mathbf{r} = G(n, \theta_1, \theta_2, \mu_1, \mu_2)$, where G is the non-deterministic mapping defined in (3.9). This vector plays the role of an *ephemeral key*.
- `ENCRYPT`: takes as input the vector $\mathbf{x}$, generated by `ENCODE`, and outputs $\mathbf{c} = \mathbf{r} + \mathbf{x}P$, where $\mathbf{r}$ is the output of `EPHEMERAL_KEY`.

Finally, we describe the decryption procedure, split into the following algorithms:

- `ERROR_VECTOR`: builds the rounding difference vector $\mathbf{d}$ defined in (3.26) and outputs the vector $\mathbf{e}'$ as follows: for all $j = 0, \cdots, n-1$, if $\mathbf{d}_j > 0$, then $\mathbf{e}'_j = \delta$, otherwise $\mathbf{e}'_j = -\delta$. As already mentioned in the previous chapter, $\mathbf{e}' = \lceil \mathbf{e} \rfloor$.
- `DECRYPT`: computes $\mathbf{x} = (\lceil \mathbf{c}' S^{-1} \rfloor - \mathbf{e}')U^{-1}$.
- `DECODE` decodes the vector $\mathbf{x}$ in order to retrieve the original binary message.

The correctness of the decryption procedure comes immediately from the correctness of the inversion procedure for our trapdoor function.

Note that the secret key consists of three parts: S^{-1}, U^{-1} and δ. Apparently, neither of these parts isolated allows full decryption of a ciphertext. For example, if the value of δ leaks to an attacker, he will still not be able to decrypt any ciphertext. The same is true if one of the secret matrices leak. Hence, our cryptosystem seems to

be suitable for scenarios of secret sharing, where the secret key must be distributed amongst a group of participants, in such a way that individual parts of the secret key are of no use on their own.

We also note that our encryption scheme resembles GGH [16], in the sense that a message is encoded into a lattice point and *perturbed* by a random vector (see Figure 4.1).

Figure 4.1: GGH encryption: the lattice point encoding the message is the closest to the ciphertext.

However, in our case the lattice vector into which the message is encoded, namely $\mathbf{x}P$, is not supposed to be the vector of $\mathcal{L}(P)$ closest to the ciphertext $\mathbf{c}$ (see Figure 4.2).

Figure 4.2: Our encryption: the lattice point encoding the message is not the closest to the ciphertext.

4.1 Implementation Remarks

Matrix inversion is a crucial part of the cryptosystem presented in this chapter, but it may also lead to round-off errors, which could cause a ciphertext to be incorrectly decrypted. In order to avoid these errors, we implemented decryption in a slightly different way. Before we proceed, let us present some useful results.

Theorem 4.1 (Cramer's Rule) *Consider the linear system* $\mathbf{x}A = \mathbf{b}$*, where* A *is a nonsingular real matrix. Then, for all* $k = 0, 1, \cdots, n-1$*, the following holds:*

$$\mathbf{x}_k = \frac{\det(A_k^{\mathbf{b}})}{\det(A)}, \tag{4.1}$$

where $A_k^{\mathbf{b}}$ *is the matrix obtained by replacing row* A_k *by the vector* $\mathbf{b}$*.*

The proof of theorem 4.1 can be found in [25].

Lemma 4.1 *Let* A *be a nonsingular* $n \times n$ *integer matrix. There is a unique integer matrix* A' *such that* $AA' = A'A = \det(A)I_n$*, where* I_n *is the identity matrix of order* n*.* A' *is said to be the integer inverse of* A*.*

Proof: Since A is an integer matrix, by Cramer's rule, each entry of A^{-1} is a rational number of the form $k_{i,j}/\det(A)$, where $k_{i,j} \in \mathbb{Z}$. Hence, the matrix $A' = \det(A)A^{-1}$ has integer entries, and $A'A = AA' = \det(A)I_n$.

□

The previous lemma allows us to *invert* an integer matrix without having to deal with floating point numbers. Although we pay the price of storing the

determinant, this strategy avoids round-off errors. We present below a modified version of the decryption algorithm, using integer inverses of the secret matrices. Recall that the ciphertext is given by $\mathbf{c} = \mathbf{r} + \mathbf{x}P$.

- Compute
$$\mathbf{d}' = \mathbf{c}S' - \det(S)\lceil \frac{1}{\det(S)}\mathbf{c}S' \rfloor. \tag{4.2}$$
Note that $\mathbf{d}' = \det(S)(\mathbf{c}S^{-1} - \lceil \mathbf{c}S^{-1} \rfloor) = \det(S)\mathbf{d}$, where $\mathbf{d}$ is the rounding difference vector defined in (3.26).

- Build the vector $\mathbf{e}''$ such that
$$\frac{\mathbf{d}'_j}{\det(S)} > 0 \Rightarrow \mathbf{e}''_j = \delta \tag{4.3}$$
and
$$\frac{\mathbf{d}'_j}{\det(S)} < 0 \Rightarrow \mathbf{e}''_j = -\delta. \tag{4.4}$$
Note that $\mathbf{e}'' = \mathbf{e}' = \lceil \mathbf{e} \rfloor$, which is the rounded inversion error vector defined in (3.12).

- Since $S' = \det(S)S^{-1}$, the final step of the decryption procedure consists of computing the vector
$$\left(\lceil \frac{1}{\det(S)}\mathbf{c}S' \rfloor - \mathbf{e}'' \right) U' = \det(U) \left(\lceil \mathbf{c}S^{-1} \rfloor - \lceil \mathbf{e} \rfloor \right) U^{-1} = \det(U)\mathbf{x}. \tag{4.5}$$

With this modification, the secret key is the set $\{U', \det(U), S', \det(S), \delta\}$. As already mentioned, we pay the price of storing the determinants, but since one of them is ± 1, this is an acceptable cost in order to avoid decryption errors.

Closing this section, we propose a straightforward encoding procedure that can be adopted in our cryptosystem. Recall that the binary message is encoded into the vector $\mathbf{x}$, and each entry of $\mathbf{x}$ lies in the set $\mathcal{Z}^{\sigma}$. We consider as input a binary string b.

1. Split b into blocks of $n\lfloor \log_2 \sigma \rfloor$ bits each. Each block yields a vector $\mathbf{x}$, which will be encrypted separately. Apply some padding procedure to the last block, if necessary. It is convenient to choose σ as a power of 2.

2. Given each block from the previous item, split it into n parts of $\lfloor \log_2 \sigma \rfloor$ bits each, denoted by $b_0, b_1, \cdots, b_{n-1}$.

3. Set $\mathbf{x}_j$ as the integer whose binary representation is given by b_j, for all $j = 0, 1, \cdots, n-1$.

4.2 Common Attacks

In this section, we assess the security of the proposed cryptosystem against common attacks, establishing the countermeasures that must be taken in order to avoid them.

4.2.1 Exhaustive Search

The first and most straightforward attack we consider is exhaustive search on the key space, in which the attacker literally tries to *guess* the secret key. We conceive a threat model in which the adversary has access to the values of the parameters $\gamma, \tau, \alpha, \beta$. Although these parameters are not public, the ranges from which their values are chosen will be eventually known by an attacker. Hence, such threat model is reasonable.

In order to measure the hardness of this attack, we must compute the size of the search space for each of the secret matrices, taking into account that the attacker only needs to guess two of them.

For each γ, the search space for the matrix $S = \gamma I + Q$ is equal to the search space for the matrix Q, whose size is given by $(\tau + 1)^{n^2}$. For the matrix E, the size of the search space is given by $(\alpha + \beta + 1)^{n^2}$. Finally, for the matrix U, we must take into account the search spaces for the matrices T and T', whose sizes are $2^n(2\omega + 1)^{n^2/2-n}$ each, and the search spaces for R and C, which have sizes equal to $n!$ each. Thus, the attacker has the following possibilities:

- Guess U and S: the total size of the search space is given by

$$2^n(n!)^2(\tau + 1)^{n^2}(2\omega + 1)^{n^2-2n}. \tag{4.6}$$

- Guess S and E: the total size of the search space is given by

$$(\tau + 1)^{n^2}(\alpha + \beta + 1)^{n^2}. \tag{4.7}$$

- Guess U and E: the total size of the search space is given by

$$2^n(n!)^2(\alpha + \beta + 1)^{n^2}(2\omega + 1)^{n^2-2n}. \tag{4.8}$$

In any of the three cases above, the size of the search space is polynomial in the parameters $\alpha, \beta, \tau, \omega$, while exponential or factorial in n. Hence, the better way to avoid exhaustive search attacks consists of (as we intuitively already expected) choosing a sufficiently large dimension n.

Also note that we did not consider the parameter δ, which is also part of the secret key. As will be shown in the next section, the search space for this parameter is quite small, when compared to the search space for the secret matrices. However, in a real world scenario, it would have to be taken into account.

4.2.2 Lattice reduction

As we already mentioned, the encryption method of our cryptosystem is similar to the GGH scheme. Hence, we must consider the idea of applying Babai's algorithm [26] to the ciphertext $\mathbf{c} = \mathbf{r} + \mathbf{x}P$. The attack works as follows:

- The attacker applies some lattice reduction technique, such as LLL [27] or BKZ [28], to the public matrix P, obtaining a reduced basis $P' = U'P$, where U' is a unimodular matrix;

- The attacker computes $\lceil \mathbf{c}P'^{-1} \rfloor = \lceil \mathbf{r}P'^{-1} \rfloor + \mathbf{x}U'^{-1}$;

- If the vector $\mathbf{r}$ is sufficiently short, the attacker may expect that the vector above is equal to $\mathbf{x}U'^{-1}$. Multiplying by P', he obtains $\mathbf{x}P$ and solves a linear system to obtain $\mathbf{x}$.

This attack may work whenever $\mathbf{r}$ is a sufficiently short vector with respect to $\mathcal{L}(P)$, so that $\mathbf{c}$ becomes sufficiently close to the lattice point $\mathbf{x}P$. Our experiments suggest that, in order to prevent $\mathbf{x}P$ from being the lattice vector closest to $\mathbf{c}$, it is sufficient to choose $\mathbf{r}$ with length greater than the length of the shortest vector in P. This conjecture was confirmed for all sets of parameters (described in the next section). As a matter of fact, even this lower bound can be considered loose, because lattice reduction attacks failed in some cases where $\mathbf{r}$ was smaller than the shortest vector in P. We emphasize that the choice of this lower bound for the length of the vector $\mathbf{r}$ was heuristic, based on purely empirical results.

4.2.3 Shortening the Perturbation Vector

The previous attack does not work basically because the vector $\mathbf{r}$ is too large, which means that the vector $\mathbf{c}$ is not close to $\mathbf{x}P$. However, an attacker may try to *shorten* the vector $\mathbf{r}$, in order to obtain a vector which is closer to $\mathbf{x}P$.

In fact, let $\mathbf{c} = \mathbf{r} + \mathbf{x}P$ be the ciphertext, and consider an integer vector $\mathbf{s}$, with entries in $[\mu_1, \mu_2] \cup [\theta_1, \theta_2]$, such that $\mathbf{r}_j\mathbf{s}_j > 0$ for all $j = 0, \cdots, n-1$. It is clear that the vector $\mathbf{r}' = \mathbf{r} - \mathbf{s}$ has smaller entries than the vector $\mathbf{r}$. The attack consists of applying Babai's algorithm to the vector $\mathbf{c} - \mathbf{s}$, which is closer to the vector $\mathbf{x}P$ than $\mathbf{c}$.

This attack will succeed if the positive (respectively negative) entries of $\mathbf{s}$ match the positive (respectively negative) entries of $\mathbf{r}$, which gives the attacker $\mathcal{O}(2^n)$ possibilities to try.

We raise a natural question: how does the attacker know whether the right vector $\mathbf{s}$ was chosen? After applying Babai's algorithm to $\mathbf{c}-\mathbf{s}$, one expects to obtain the vector $\mathbf{x}$. Therefore, one may check whether the answer is a vector with integer entries in the set $\mathcal{Z}^\sigma$. Our experiments suggest that this attack yields a unique vector with entries in the range $[0, \sigma]$. Curiously, if a random integer matrix is used instead of the unimodular U to build the public matrix P, this attack produces more than one vector with entries within this range. The attacker has no way of knowing which one of them is the right one, because the encryption is randomized.

4.3 Performance Analysis

In this section, we prove the existence of parameters satisfying conditions (3.1) to (3.8) by presenting four possible sets of such parameters. We provide an analysis of the key size and the hardness of the aforementioned attacks for each one of these sets. All values suggested here were tested using the NTL C++ Library [29], running on Linux Mint 17 (64 bits) with an AMD E-350 1600MHz processor and 4GB of RAM.

The choices for $\sigma, \delta, \gamma, \tau, \alpha, \beta, \omega, \epsilon_1, \epsilon_2$ were heuristically determined in order to produce suitable values for $\mu_1, \mu_2, \theta_1, \theta_2$ and generate valid $M_{\gamma,\tau}$-matrices satisfying conditions (3.1) and (3.2). There must be a fine tuning between the parameters. Bad choices may result in:

1. Invalid $M_{\gamma,\tau}$-matrices (whose inverses do not satisfy conditions (3.1) and (3.2));
2. Incorrect values for $\mu_1, \mu_2, \theta_1, \theta_2$ (with $\theta_1 > \theta_2$, for example);
3. Ciphertexts vulnerable to lattice reduction attacks (if the vector $\mathbf{r}$ is too short).

From now on, let rand(x) be a function that outputs a random integer from the interval $[0, x[$. The values of the parameters $\mu_1, \mu_2, \theta_1, \theta_2$ were computed as follows:

- $\mu_1 = \lceil \dfrac{-\gamma^2(2\delta+1)}{2(\gamma(1+\epsilon_1)+\tau(n-1)(1+\epsilon_2))} \rceil + n\sigma\alpha + \text{rand}(64)$.
- $\theta_1 = \gamma\delta + n\sigma\alpha - \lfloor \dfrac{\tau(n-1)(\mu_1 - n\sigma\alpha)(1+\epsilon_2)}{\gamma} \rfloor + \text{rand}(64)$.
- $\theta_2 = \lfloor \dfrac{\gamma^2(2\delta+1)}{2(\gamma(1+\epsilon_1)+\tau(n-1)(1+\epsilon_2))} \rfloor - n\sigma\beta - \text{rand}(64)$.

- $\mu_2 = -\gamma\delta - n\sigma\beta - \lceil \frac{\tau(n-1)(\theta_2 + n\sigma\beta)(1+\epsilon_2)}{\gamma} \rceil - \text{rand}(64)$.

In tables 4.1 and 4.2 we present four possible sets of parameters. In all cases, we adopted $\delta = 256 + \text{rand}(256)$ and $\omega = 2$. In all sets, the vector $\mathbf{r}$ is large enough to avoid the lattice reduction attack described in the previous section.

Set 1		Set 2	
Parameter	Value	Parameter	Value
α	2	α	2
β	3	β	3
γ	$n^3 + \text{rand}(n)$	γ	$n^4 + \text{rand}(n)$
τ	1	τ	1
σ	256	σ	256
ϵ_1	10^{-4}	ϵ_1	10^{-3}
ϵ_2	10^{-2}	ϵ_2	10^{-2}

Table 4.1: Sets of parameters 1 and 2.

Set 3		Set 4	
Parameter	Value	Parameter	Value
α	$n + \text{rand}(n+1)$	α	$n + \text{rand}(n+1)$
β	$\alpha + \text{rand}(n+1)$	β	$\alpha + \text{rand}(n+1)$
γ	$n^4 + \text{rand}(n^2)$	γ	$n^5 + \text{rand}(n^3)$
τ	$n/2 + \text{rand}(n/2+1)$	τ	$n/2 + \text{rand}(n/2+1)$
σ	256	σ	256
ϵ_1	10^{-5}	ϵ_1	10^{-6}
ϵ_2	10^{-4}	ϵ_2	10^{-4}

Table 4.2: Sets of parameters 3 and 4.

Sets 1 and 2 yield a search space of size $\Omega(2^{\mathcal{O}(n^2)})$, while for sets 3 and 4 the search space size is $\Omega(n^{\mathcal{O}(n^2)}2^{\mathcal{O}(n)})$. We suggest 128 as the minimum value for the security parameter n. The maximum value we tested was $n = 256$.

Table 4.3 shows the public key size, table 4.4 shows the average key generation time (in seconds), table 4.5 shows the average encryption time (in seconds) and

table 4.6 shows the average decryption time (in seconds) for each parameter set, considering the minimum and maximum security parameter.

	Public key size			
Dimension (n)	Set 1	Set 2	Set 3	Set 4
128	54 KB	65 KB	65 KB	78 KB
256	249 KB	311 KB	312 KB	375 KB

Table 4.3: Public key sizes (in kB) for each one of the parameter sets, considering the minimum and maximum security parameter.

	Key generation time			
Dimension (n)	Set 1	Set 2	Set 3	Set 4
128	$8.09s$	$10.43s$	$11.48s$	$14.29s$
256	$123.04s$	$169.21s$	$187.58s$	$241.72s$

Table 4.4: Key generation time (in seconds) for each one of the parameter sets, considering the minimum and maximum security parameter.

	Encryption time			
Dimension (n)	Set 1	Set 2	Set 3	Set 4
128	$0.00378s$	$0.00388s$	$0.00382s$	$0.00386s$
256	$0.01497s$	$0.01611s$	$0.01598s$	$0.01492s$

Table 4.5: Encryption time (in seconds) for each one of the parameter sets, considering the minimum and maximum security parameter.

	Decryption time			
Dimension (n)	Set 1	Set 2	Set 3	Set 4
128	$0.02074s$	$0.02474s$	$0.02479s$	$0.03095s$
256	$0.15116s$	$0.19747s$	$0.19911s$	$0.33761s$

Table 4.6: Decryption time (in seconds) for each one of the parameter sets, considering the minimum and maximum security parameter.

5 LATTICE DEFORMATIONS

"I want the pain. It's how I learn." (Caprica-Six)

In this chapter, we introduce the problem of lattice deformations, which comes from a geometrical approach to the problem of exposing the secret key of our cryptosystem.

Problem 5.1 (Lattice Deformation Problem - LDP) *We take as parameters of the problem three subsets $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathbb{Z}^{n \times n}$. The input consists of a lattice basis P. The goal is to find $R \in \mathcal{A}$ and $S \in \mathcal{B}$ such that $P - RS \in \mathcal{C}$.*

Consider three subsets $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathbb{Z}^{n \times n}$. Given a lattice basis $S \in \mathcal{B}$, an $\mathcal{A}, \mathcal{C}$-deformation of S consists of two steps:

1. Obtain a basis for a sublattice of $\mathcal{L}(S)$ given by RS, where $R \in \mathcal{A}$;
2. Add a *deformation matrix* $E \in \mathcal{C}$, obtaining a basis $P = RS + E$.

In other words, the goal of LDP is to find a lattice basis S and a matrix R such that P is the result of an $\mathcal{A}, \mathcal{C}$-deformation of S. The term *deformation* was chosen because, from a geometrical standpoint, the grid of points that constitute the sublattice of $\mathcal{L}(S)$ has its shape changed (deformed) by the vectors that are added to the basis of this sublattice (see Figure 5.1).

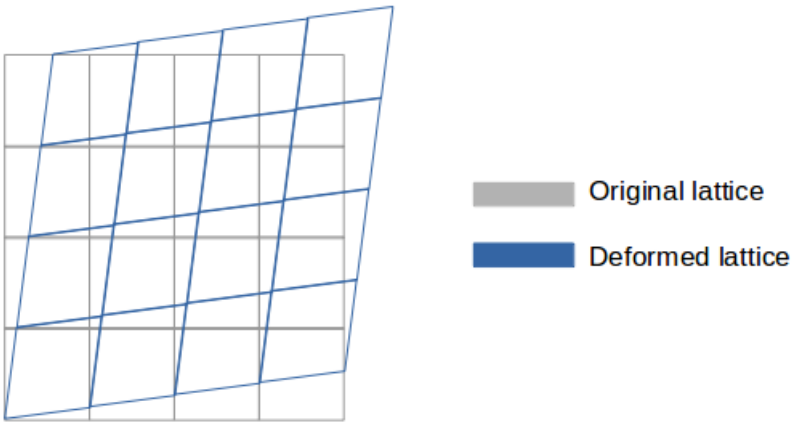

Figure 5.1: Lattice deformation.

An instance of LDP with parameters $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and input P is denoted by $\text{LDP}^{\mathcal{A},\mathcal{B},\mathcal{C}}(P)$. A pair of integer matrices (R, S) is a solution for $\text{LDP}^{\mathcal{A},\mathcal{B},\mathcal{C}}(P)$ if $R \in \mathcal{A}$, $S \in \mathcal{B}$ and $P - RS \in \mathcal{C}$. We say that P is a *viable input* if such a solution exists.

Note that the formulation of LDP immediately yields a decision version, which we denote by dLDP and state as follows:

Problem 5.2 (Decisional Lattice Deformation Problem - dLDP) *We take as parameters of the problem three subsets $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathbb{Z}^{n\times n}$. The input consists of a lattice basis P. The goal is to decide whether there is an integer matrix $R \in \mathcal{A}$ and a lattice basis $S \in \mathcal{B}$ such that $P - RS \in \mathcal{C}$.*

Obviously, a solution for LDP is a certificate for a YES instance of dLDP. We employ the following special notation, for a given positive integer n:

- $\mathcal{I}$ will denote the set of integer $n \times n$ matrices;

- $\mathcal{M}_\gamma^\tau$ will denote the set of all $n \times n$ $M_{\gamma,\tau}$-matrices;
- $\mathcal{U}$ will denote the set of all n-dimensional unimodular matrices;
- $\mathcal{E}_\alpha^\beta$ will represent the set of all integer $n \times n$ matrices with entries chosen from the set $\mathcal{Z}_\alpha^\beta$, for given positive integers α, β.

In order to analyze our cryptosystem, we consider instances of $\text{LDP}^{\mathcal{A},\mathcal{B},\mathcal{C}}$ with $\mathcal{A} = \mathcal{U}$, $\mathcal{B} = \mathcal{M}_\gamma^\tau$ and $\mathcal{C} = \mathcal{E}_\alpha^\beta$. As already mentioned in the previous chapter, we work under the assumption that the adversary has access to the values of the parameters $\gamma, \tau, \alpha, \beta$. Our goal is to establish the equivalence between exposing the secret key of our cryptosystem and solving LDP. For simplicity, we denote the problem of exposing the secret key by ESK.

1. $\text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta} \propto ESK$
   (a) Let P be any viable input of $\text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta}$. We map this input to a public matrix of our cryptosystem. In fact, P itself is a public matrix for some instance of a public key.
   (b) Assume the existence of an oracle that, given a public matrix and the parameters $\gamma, \tau, \alpha, \beta$, returns the secret matrices U and S.
   (c) Feed the oracle with P and get the secret matrices U, S.
   (d) The pair (U, S) is a solution for $\text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta}(P)$.
2. $ESK \propto \text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta}$
   (a) Let P be a public matrix from a generic instance of a public key. We map P to a viable input of $\text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta}$. Because of the way P is built, it is also one such viable input.
   (b) Assume the existence of an oracle that solves $\text{LDP}^{\mathcal{U},\mathcal{M}_\gamma^\tau,\mathcal{E}_\alpha^\beta}$, for any viable input.

(c) Feed the oracle with P and get a solution pair (U, S).

(d) This solution pair serves as a pair of secret matrices. The parameter δ can be found by exhaustive search (since the space of all possible choices for δ will eventually be publicly known as well).

Hence, by assuming the hardness of $\mathrm{LDP}^{\mathcal{U},\mathcal{M}^{\tau}_{\gamma},\mathcal{E}^{\beta}_{\alpha}}$, we may conclude that finding the secret matrices from the public matrix is a hard task. But we must find a connection between LDP and some other problem, which is already known to be hard.

Let A be an n-dimensional lattice basis. According to Minkowski's convex body theorem [30], any convex symmetric body of volume greater than $2^n|\det(A)|$ contains a non-zero lattice point. If we consider the n-dimensional sphere $\mathcal{S}$, centered at the origin, of radius $\sqrt{n}|\det(A)|^{1/n}$, we see that it contains the cube

$$\mathcal{C} = [-|\det(A)|^{1/n}, |\det(A)|^{1/n}]^n, \tag{5.1}$$

which means that $\mathrm{vol}(\mathcal{S}) > \mathrm{vol}(\mathcal{C}) = 2^n|\det(A)|$. Hence, $\mathcal{S}$ contains a non-zero lattice point. Denoting by $\lambda_1(\mathcal{L}(A))$ the length of the shortest vector in $\mathcal{L}(A)$, we conclude that

$$\lambda_1(\mathcal{L}(A)) \leq \sqrt{n}|\det(A)|^{\frac{1}{n}}. \tag{5.2}$$

Using the exact formula for the volume of $\mathcal{S}$, we obtain a tighter bound, which is better only by a constant factor, given by

$$\lambda_1(\mathcal{L}(A)) \leq \sqrt{\frac{n}{\pi e}}|\det(A)|^{\frac{1}{n}}. \tag{5.3}$$

It is known that the upper bound given by (5.3) can be arbitrarily loose. For instance, consider the lattice generated by the matrix

$$\begin{pmatrix} \epsilon & 0 \\ 0 & 1/\epsilon \end{pmatrix}, \tag{5.4}$$

for some small $\epsilon > 0$. Minkowski's bound yields $\sqrt{2/\pi e}$, however the vector $(\epsilon, 0)$ can be arbitrarily smaller than that. On the other hand, for all $n > 0$, there is an

n-dimensional lattice L such that $\lambda_1(L) > c\sqrt{n}|\det(L)|^{\frac{1}{n}}$ for some constant $c > 0$ [31]. As a consequence, $\mathcal{O}(\sqrt{n})|\det(L)|^{\frac{1}{n}}$ is the best upper bound for the length of the shortest vector of any n-dimensional lattice. In other words, Minkowski's bound cannot be improved beyond small constant factors.

Given a lattice, if its Minkowski's bound is sufficiently tight, then no vectors but the shortest one lie within it (see Figure 5.2).

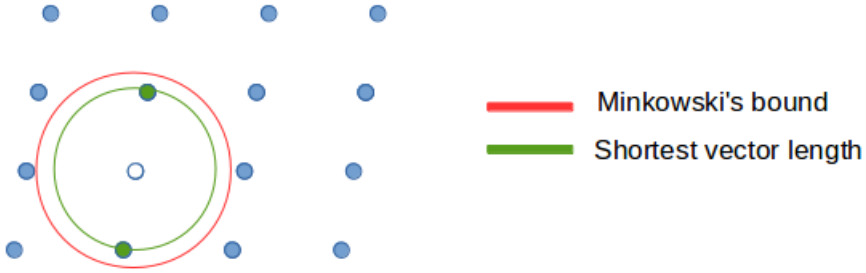

Figure 5.2: Tight Minkowski's bound. In this case, only the shortest lattice vectors (green points) lie within this bound.

On the other hand, if this bound is loose, then even a vector within it can be larger than the shortest lattice vector (see Figure 5.3). Therefore, in the general case, finding a vector within Minkowski's bound does not mean finding the shortest vector in the lattice. Nevertheless, there is no known efficient algorithm which can always find a vector within Minkowski's bound [32].

Under the assumption that finding a vector within Minkowski's bound is hard in the general case, we show that it is unlikely to exist a *universal* algorithm for LDP, i.e., an efficient algorithm that solves LDP for any family of subsets $\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$. If such algorithm exists, then we have an efficient method to find vectors within Minkowski's bound. In fact, assume we have access to an oracle that solves LDP for any family of three subsets of integer matrices.

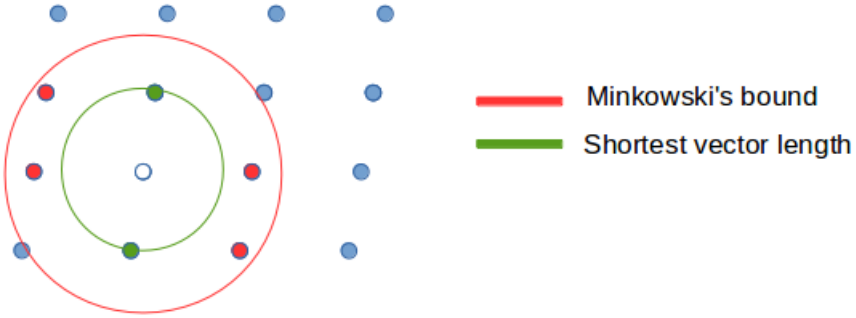

Figure 5.3: Lattice with loose Minkowski's bound. There are vectors within this bound (red points) which are larger than the shortest lattice vectors (green points).

1. Let P be an n-dimensional lattice basis.

2. We consider P as the input for an instance of $LDP^{\mathcal{A},\mathcal{B},\mathcal{C}}$, where $\mathcal{A} = \mathcal{U}$, $\mathcal{C}$ is the set that contains only the null matrix and

$$\mathcal{B} = \left\{ S \in \mathbb{Z}^{n \times n} \mid ||S_0|| \leq \sqrt{n/\pi e} |\det(P)|^{\frac{1}{n}} \right\} \tag{5.5}$$

   where S_0 is the first row of S. We claim that P is a viable input. In fact, since $\mathcal{L}(P)$ contains a vector within Minkowski's bound, we may choose other $n-1$ linearly independent vectors of $\mathcal{L}(P)$ and obtain a basis S such that $P = US$ for some unimodular matrix U, where S_0 (possibly after reordering the vectors) satisfies Minkowski's bound.

3. Given access to the oracle that solves LDP, we find a basis $S' \in \mathcal{B}$ and a matrix $U \in \mathcal{U}$ such that $P = US$. In other words, we find a basis for $\mathcal{L}(P)$ which contains a vector within Minkowski's bound.

Although we have shown the unlikelihood of an efficient universal algorithm for LDP, the question remains open as to whether there is an efficient algorithm to solve LDP for a particular family of subsets $\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$. Under the general assumption that solving LDP is hard, we conclude that exposing the secret matrices of the proposed cryptosystem is also a hard task.

6 A VARIANT WITH BETTER KEY SIZES

"All the pieces are falling into place." (Virtual Six)

Although the cryptosystem proposed in Chapter 4 offers good levels of security against known attacks, it is not optimal in terms of the key size, mostly due to the parameter γ, which is $O(n^3)$ in the best case. In this chapter, we introduce a variant of our cryptosystem, which offers much smaller keys by working with polynomials instead of matrices. As a consequence, encryption and decryption are also sped up.

Before we proceed, we present a brief review on the theory of polynomials and resultants. For an in-depth treatise on the subject, [33] can be consulted.

6.1 Polynomials Over The Integers

Let $\mathbb{Z}$ be the ring of integers. A polynomial in a single variable over $\mathbb{Z}$ is a sum of the form

$$a(x) = \sum_{j=0}^{d} a_j x^j, \tag{6.1}$$

where $a_j \in \mathbb{Z}$ for all j and $a_d \neq 0$. The integers a_j are the coefficients of the polynomial, and a_d is its *leading coefficient*. The number d is the degree of the polynomial, denoted by $\deg(a)$. Constant polynomials have degree 0, except for the null polynomial, which has no degree. If the leader coefficient is 1, we say that $a(x)$ is *monic*. As usual, the ring of polynomials with integer coefficients will be denoted by $\mathbb{Z}[x]$.

To avoid ambiguity, the coefficients will always be represented by the same letter that represents the polynomial, subscripted from 0 to its degree. For example, if $a(x)$ is a polynomial of degree n, there should be no doubt that its leading coefficient is a_n. Furthermore, whenever it is clear that $a(x)$ is a polynomial, we shall denote it simply by a.

Let $q \in \mathbb{Z}[x]$ be a monic polynomial of degree n. For all $a \in \mathbb{Z}[x]$ there is a unique pair of polynomials $t, b \in \mathbb{Z}[x]$ such that

$$a = tq + b, \tag{6.2}$$

where $0 \leq \deg(b) < n$ or $b = 0$.

Let $q \in \mathbb{Z}[x]$ be a monic polynomial of degree n. Define the relation $\equiv_q$, for all $a, b \in \mathbb{Z}[x]$, as follows:

$$a \equiv_q b \Leftrightarrow a - b = tq, \text{ for some } t \in \mathbb{Z}[x]. \tag{6.3}$$

We observe that $\equiv_q$ is an equivalence relation. In fact:

1. $\equiv_q$ is reflexive, because $a - a = 0 = 0 \cdot q$, with $t = 0$;
2. $\equiv_q$ is symmetric, because whenever $a - b = tq$, we have $b - a = -tq$;
3. Finally, if $a - b = t_1 q$ and $b - c = t_2 q$, then $a - c = (t_1 + t_2)q$, which shows that $\equiv_q$ is transitive.

Whenever $a \equiv_q b$, it is usual to write $a \equiv b \pmod{q}$. Essentially, it means that a and b leave the same remainder when divided by q. In particular, if $\deg(b) < \deg(q)$, then b is itself the remainder from the division of a by q, which will be denoted by $a \bmod q$.

The equivalence class of $a \in \mathbb{Z}[x]$ under $\equiv_q$ is the set

$$\overline{a} = \{b \in \mathbb{Z}[x] \mid a \equiv b \pmod{q}\}. \tag{6.4}$$

The set of all equivalence classes under $\equiv_q$ is the quotient ring $\mathbb{Z}[x]/(q)$. By (6.2), for all $a \in \mathbb{Z}[x]$ there is a unique polynomial $b \in \mathbb{Z}[x]$ of degree less than n such that $a \equiv b \pmod{q}$. Hence, $\mathbb{Z}[x]/(q)$ is fully represented by the equivalence classes of polynomials with degree less than n, together with the equivalence class of the null polynomial. Consequently, the elements of $\mathbb{Z}[x]/(q)$ are in a one-to-one correspondence with the elements of the set

$$\{b \in \mathbb{Z}[x] \mid 0 \leq \deg(b) < n\} \cup \{0\}, \tag{6.5}$$

which are in a one-to-one correspondence with the elements of $\mathbb{Z}^n$. In fact, for all a in the set defined in (6.5), there is a unique vector in $\mathbb{Z}^n$, denoted by $[a]$, such that $[a]_k = a_k$, for all $k = 0, 1, \cdots, n-1$. In a more general way, to any polynomial $a \in \mathbb{Z}[x]$ of degree d corresponds a vector $[a] \in \mathbb{Z}^k$, for any $k > d$, such that $[a]_j = a_j$, for $j \leq d$, and $[a]_j = 0$ for all $j > d$. For example, if $a = 2 - 3x$, then $[a] = (2, -3)$ in $\mathbb{Z}^2$, and $[a] = (2, -3, 0, 0)$ in $\mathbb{Z}^4$. The vector $[a]$ is said to be the **coefficient vector** of a.

6.1.1 Resultants

Given two polynomials $a, b \in \mathbb{Z}[x]$, with $\deg(a) = m$ and $\deg(b) = n$, the resultant of a and b is given by

$$\mathrm{Res}(a, b) = a_m^n b_n^m \prod_{i,j} (\alpha_i - \beta_j), \tag{6.6}$$

where α_i and β_j are the roots (over $\mathbb{C}$) of a and b respectively. We note that the resultant is zero iff a and b have a common root. Also note that, if a and b are both monic, then we may write

$$\mathrm{Res}(a, b) = \prod_{i,j} (\alpha_i - \beta_j). \tag{6.7}$$

Taking into account that

$$a(x) = \prod_{i=1}^{m}(x - \alpha_i) \tag{6.8}$$

and

$$b(x) = \prod_{j=1}^{n}(x - \beta_j), \tag{6.9}$$

we see that

$$\mathrm{Res}(a, b) = \prod_{i=1}^{m} b(\alpha_i) = (-1)^{mn} \prod_{j=1}^{n} a(\beta_j). \tag{6.10}$$

We may compute the resultant as a function of the coefficients of a and b, without having to directly compute all the roots of these polynomials.

Definition 6.1 *Given two polynomials $a, b \in \mathbb{Z}[x]$ of degrees m and n respectively, the* **Sylvester Matrix** *of a and b is given by*

$$S_{a,b} = \begin{pmatrix} a_0 & a_1 & \cdots & \cdots & \cdots & a_m & & & \\ & a_0 & a_1 & \cdots & \cdots & \cdots & a_m & & \\ & & \cdots & & & & & & \\ & & & a_0 & a_1 & \cdots & \cdots & \cdots & a_m \\ b_0 & b_1 & \cdots & \cdots & \cdots & b_n & & & \\ & b_0 & b_1 & \cdots & \cdots & \cdots & b_n & & \\ & & \cdots & & & & & & \\ & & & \cdots & & & & & \\ & & & & b_0 & b_1 & \cdots & \cdots & b_n \end{pmatrix} \tag{6.11}$$

Its first n rows contain the coefficients of a, and its m last rows contain the coefficients of b.

Lemma 6.1 *Given two polynomials $a, b \in \mathbb{Z}[x]$ of degrees m and n respectively, the following holds:*

$$\mathrm{Res}(a, b) = \det(S_{a,b}). \tag{6.12}$$

Proof: We know that $\mathrm{Res}(a,b) = 0$ iff a and b have a common root, i.e, iff they have a nontrivial g.c.d.. In other words, iff there are polynomials h, f, g such that $a = fh$ and $b = gh$, with $\deg(f) \leq m-1$ and $\deg(g) \leq n-1$. In this case, $ga - fb = 0$. This yields a linear system with $m+n$ homogeneous equations, where the unknowns are the coefficients of f and g. The matrix of this system is exactly $S_{a,b}$, and it has a nontrivial solution iff the determinant vanishes.

On the other hand, by (6.11), $\mathrm{Res}(a,b)$ is an expression of degree n in the a_i's and m in the b_j's, and so is $\det(S_{a,b})$. Viewing the coefficients as functions of the roots in the expression of the determinant, we conclude that every factor of the resultant is also a factor of the determinant. Provided that each factor of the resultant has multiplicity 1, it must be equal to the determinant up to some constant. But looking at the term $a_0^n b_n^m$ from (6.6), we conclude that this constant must be 1. □

Assume that $q \in \mathbb{Z}[x]$ is monic of degree n, and $a \in \mathbb{Z}[x]$ has degree $m < n$. Instead of computing a determinant of order $m+n$, we may simply compute a determinant of order n to obtain the resultant of a and q. Before we proceed, we present the following result.

Lemma 6.2 *Let $q \in \mathbb{Z}[x]$ be a monic polynomial of degree n. Given any $a \in \mathbb{Z}[x]$ such that $\deg(a) = m < n$, there is a unique $a' \in \mathbb{Z}[x]$, of degree less than n, such that $aa' \equiv \mathrm{Res}(a,q) \pmod{q}$. The polynomial a' is said to be the integer inverse of a modulo q.*

Proof: Consider the equation $ra + sq = 1$ over $\mathbb{Q}[x]$, where the unknowns are the coefficients of r and s, with $\deg(r) \leq n-1$ and $\deg(s) \leq m-1$. This yields the linear system

$$\mathbf{u}S_{a,q} = \mathbf{e}^{(0)}, \tag{6.13}$$

where $\mathbf{u} \in \mathbb{Q}^{m+n}$ is the vector whose first n entries represent the coefficients of r, and the last m entries represent the coefficients of s. If $S_{a,q}$ is nonsingular, Cramer's Rule guarantees that the system has a unique solution such that $\mathbf{u}_j = k_j/\det(S_{a,q})$, with $k_j \in \mathbb{Z}$ for all j. Hence, there is a vector $\mathbf{u}' = \det(S_{a,q})\mathbf{u} \in \mathbb{Z}^{m+n}$ such that

$$\mathbf{u}' S_{a,q} = \mathrm{Res}(a, q)\mathbf{e}^{(0)}. \tag{6.14}$$

Let us denote the concatenation of vectors by the symbol $||$. By construction, $\mathbf{u}' = [a']||[q']$, where a' and q' are integer polynomials such that $a'a + q'q = \mathrm{Res}(a, q)$, with $\deg(a') \leq n - 1$ and $\deg(q') \leq m - 1$. Therefore, $aa' \equiv \mathrm{Res}(a, q) \pmod{q}$.

On the other hand, if $\mathrm{Res}(a, q) = 0$, then a and q have a nontrivial g.c.d.. In other words, there are integer polynomials f, g, h such that $a = fh$ and $q = gh$, with $\deg(g) < n$. Hence, $ga - fq = 0$, and we see that $a' = g$. $\square$

Definition 6.2 *Let $q \in \mathbb{Z}[x]$ be monic of degree n, and $a \in \mathbb{Z}[x]$. The* **Characteristic Matrix** *of a modulo q is the $n \times n$ integer matrix $C_{a,q}$ such that each row $(C_{a,q})_k$ is the coefficient vector of $x^k a \bmod q$, for all $k = 0, 1, \cdots, n - 1$.*

For example, consider $q(x) = 1 + 2x + x^2$ and $a(x) = 2 - x$. Then we have that $xa \equiv 1 + 4x \pmod{q}$, and the characteristic matrix of a modulo q is given by

$$C_{a,q} = \begin{pmatrix} 2 & -1 \\ 1 & 4 \end{pmatrix}. \tag{6.15}$$

The following lemma establishes the relation between the resultant and the determinant of characteristic matrices. It is a key result that will allow us to build an efficient variant of the cryptosystem presented in Chapter 4.

Lemma 6.3 *Let $q \in \mathbb{Z}[x]$ be monic of degree n, and $a \in \mathbb{Z}[x]$ of degree $m < n$.*

Then

$$\mathrm{Res}(a, q) = \det(C_{a,q}). \tag{6.16}$$

Proof: By performing elementary row operations, we are going to obtain $\mathrm{C}_{a,q}$ as a submatrix of $S_{a,q}$ and show that their determinants are equal. This proof will be illustrated by an example, in order to make it easier to visualize the step-by-step procedure. In our example, we are going to consider $q(x) = 2 - x + x^2 + 2x^3 + x^4$ and $a(x) = 1 + 2x - x^2 + x^3$. Hence, we have

$$S_{a,q} = \begin{pmatrix} 1 & 2 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & -1 & 1 \\ 2 & -1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & -1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & -1 & 1 & 2 & 1 \end{pmatrix} \tag{6.17}$$

Before we proceed, we make some remarks on the structure of $S_{a,q}$:

1. Since q is monic, the $m \times m$ rightmost lower square of $S_{a,q}$ is a lower triangular matrix with unitary diagonal;

2. For each $k = 0, \cdots, n-1$, the first n rows of $S_{a,q}$ correspond to $[x^k a]$ over $\mathbb{Z}^{m+n}$;

3. For each $k = 0, \cdots, m-1$, the last m rows of $S_{a,q}$ correspond to $[x^k q]$ over $\mathbb{Z}^{m+n}$.

By (6.2), for all $k = 0, \cdots, n-1$, there is a pair of polynomials t_k, r_k such that $x^k a = t_k q + r_k$, where $0 \leq \deg(r_k) < n$. Because $\deg(x^k a) \leq m + n - 1$, we conclude that $\deg(t_k) < m$. Hence, $t_k q$ is a linear combination of $q, xq, x^2 q, \cdots, x^{m-1} q$ and, consequently, $[t_k q]$ is a linear combination of the last m rows of $S_{a,q}$. This means

that each $[r_k]$ consists of row $(S_{a,q})_k$ plus a linear combination of the last m rows of $S_{a,q}$.

Hence, we may employ the following procedure: for each $k = 0, \cdots, n-1$, replace the row $(S_{a,q})_k$ by itself minus the linear combination of the last m rows corresponding to $[t_k q]$. Let us see what happens in our example:

1. $a(x) = 0 \cdot q(x) + (1 + 2x - x^2 + x^3)$, hence row $(S_{a,q})_0$ remains unchanged;
2. $xa(x) = 1 \cdot q(x) + (-2 + 2x + x^2 - 3x^3)$, hence row $(S_{a,q})_1$ row will be replaced by itself minus row $(S_{a,q})_4$;
3. $x^2 a(x) = (-3+x)q(x) + (6 - 5x + 5x^2 + 7x^3)$, hence row $(S_{a,q})_2$ will be replaced by itself plus 3 times row $(S_{a,q})_4$ minus row $(S_{a,q})_5$, which is equivalent to replacing it by $[6 - 5x + 5x^2 + 7x^3]$;
4. Finally, $x^3 a(x) = (7 - 3x + x^2)q(x) + (-14 + 13x - 12x^2 - 9x^3)$. Therefore, row $(S_{a,q})_3$ will be replaced by itself minus 7 times row $(S_{a,q})_4$, plus 3 times row $(S_{a,q})_5$ minus row $(S_{a,q})_6$, i.e., by $[-14 + 13x - 12x^2 - 9x^3]$.

After these row operations, we obtain the matrix

$$S' = \begin{pmatrix} 1 & 2 & -1 & 1 & 0 & 0 & 0 \\ -2 & 2 & 1 & -3 & 0 & 0 & 0 \\ 6 & -5 & 5 & 7 & 0 & 0 & 0 \\ -14 & 13 & -12 & -9 & 0 & 0 & 0 \\ 2 & -1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & -1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & -1 & 1 & 2 & 1 \end{pmatrix} \tag{6.18}$$

whose determinant is equal to $\det(S_{a,q})$. Note that, in the general case, because each r_k has degree less than n, each one of the first n rows of S' corresponds to $[r_k]$ over $\mathbb{Z}^n$ followed by m zeros. In other words, the $n \times n$ leftmost upper square of S' corresponds to $C_{a,q}$.

Now observe that each one of the last m rows of S' ends with a 1 followed by 0's (except for the last row), and each of these 1's has only 0's above it. Dividing the matrix in blocks, as shown below,

$$S' = \left(\begin{array}{cccc|ccc} 1 & 2 & -1 & 1 & 0 & 0 & 0 \\ -2 & 2 & 1 & -3 & 0 & 0 & 0 \\ 6 & -5 & 5 & 7 & 0 & 0 & 0 \\ -14 & 13 & -12 & -9 & 0 & 0 & 0 \\ \hline 2 & -1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & -1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & -1 & 1 & 2 & 1 \end{array}\right) \tag{6.19}$$

and provided that S' is a block lower triangular matrix [25], its determinant is given by the product of the determinants of the upper leftmost block, which is $\det(C_{a,q})$, and the lower rightmost block, which is 1. Hence, we obtain $\det(S') = \det(C_{a,q})$. □

Note that $C_{a,q}$ is fully determined by a and q. Characteristic matrices have the following properties, for all monic polynomial $q \in \mathbb{Z}[x]$ of degree n and $a, b \in \mathbb{Z}[x]$ of degrees less than n:

Proposition 6.1 $C_{a,q} \pm C_{b,q} = C_{a \pm b,q}$.

Proof: By construction, $(C_{a,q})_j \pm (C_{b,q})_j$ is given by $[x^j a \bmod q] \pm [x^j b \bmod q] = [x^j(a \pm b) \bmod q]$, for all $j = 0, 1, \cdots, n-1$. This is exactly the row $(C_{a \pm b,q})_j$. □

Proposition 6.2 $C_{a,q} C_{b,q} = C_{ab,q}$.

Proof: Observe that

$$[x^k] C_{b,q} = [x^k b \bmod q], \tag{6.20}$$

for all $k = 0, \cdots, n-1$. Since every polynomial of degree less than n may be written as a linear combination of these x^k, we conclude that

$$[a]C_{b,q} = [ab \bmod q], \tag{6.21}$$

for all $a \in \mathbb{Z}[x]$ of degree less than n. In particular,

$$[x^k a \bmod q]C_{b,q} = [x^k ab \bmod q]. \tag{6.22}$$

But $[x^k a \bmod q]$ is the row $(C_{a,q})_k$, and $[x^k ab \bmod q]$ the row $(C_{ab,q})_k$. Hence, we conclude that $C_{a,q}C_{b,q} = C_{ab,q}$. □

Corollary 6.1 *$C_{a,q}C_{a',q} = \det(C_{a,q})I_n$, where I_n is the identity matrix of dimension n and a' is the integer inverse of a modulo q.*

Proof: This is a particular case of Proposition 2. In fact,

$$C_{a,q}C_{a',q} = C_{aa',q}. \tag{6.23}$$

As we have shown in Lemma 3, $aa' \equiv \mathrm{Res}(a, q) \pmod q$ and, by Lemma 4, $\mathrm{Res}(a, q) = \det(C_{a,q})$. □

Note that matrix $C_{a',q}$ from Corollary 1 is the integer inverse of $C_{a,q}$. The examples below illustrate the properties of characteristic matrices, for $a(x) = 1+2x$ and $b(x) = 2 - x$, with $q(x) = 1 - x + x^2$:

$$C_{a,q} = \begin{pmatrix} 1 & 2 \\ -2 & 3 \end{pmatrix}, \; C_{b,q} = \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix} \tag{6.24}$$

We have that $a + b \equiv 3 + x \pmod q$, and

$$C_{a+b,q} = \begin{pmatrix} 3 & 1 \\ -1 & 4 \end{pmatrix} = C_{a,q} + C_{b,q}. \tag{6.25}$$

On the other hand, we have that $ab \equiv 4 + x \pmod{q}$, and

$$C_{ab,q} = \begin{pmatrix} 4 & 1 \\ -1 & 5 \end{pmatrix} = C_{a,q}C_{b,q}. \tag{6.26}$$

Finally we have that $a'(x) = 3 - 2x$ and

$$C_{a,q}C_{a',q} = \begin{pmatrix} 7 & 0 \\ 0 & 7 \end{pmatrix} = 7I_2 = \det(C_{a,q})I_2. \tag{6.27}$$

We note that operations involving characteristic matrices are equivalent to operations involving polynomials modulo q. This fact will be at the centre of our construction for the variant of the proposed cryptosystem, as will be shown in the next subsection.

6.2 An Efficient Variant of the Proposed Cryptosystem

In this section, we take everything we have seen so far, put it all together and build an efficient variant of the cryptosystem previously proposed. The core idea consists of using characteristic matrices to generate the keys. Since all the operations involving such matrices are equivalent to operations with polynomials, we will be able not only to reduce the key sizes, by storing polynomials instead of matrices, but also to speed up encryption and decryption, since operations with polynomials are much faster than those involving matrices.

For key generation, we employ the following steps:

- Select a positive integer n, which is the security parameter.
- Pick up positive integers $\alpha, \beta, \gamma, \delta, \tau, \sigma, \omega$ and positive real numbers ϵ_1, ϵ_2.
- Choose a sparse random monic polynomial $q \in \mathbb{Z}[x]$ of degree n.

- Choose random polynomials $f, h \in \mathbb{Z}[x]$ of degree $n-1$, with coefficients randomly and uniformly chosen from the sets $\mathcal{Z}_{\omega}^{\omega}$ and $\mathcal{Z}_{\alpha}^{\beta}$, respectively.
- Choose a random polynomial $g \in \mathbb{Z}[x]$ of degree $n-1$ such that $\gamma - \tau \leq g_0 \leq \gamma$ and $-\tau \leq g_j \leq 0$ for all $j > 0$. Make sure that $C_{g,q}$ is an $M_{\gamma,\tau}$-matrix whose inverse satisfies (3.1) and (3.2).
- Consider the matrix $P = C_{f,q}C_{g,q} + C_{h,q}$. From the properties of characteristic matrices seen in the previous section, we have that $P = C_{fg+h,q}$. Hence, we build the polynomial
$$p = fg + h \pmod{q}. \tag{6.28}$$
- Choose integer numbers $\theta_1, \theta_2, \mu_1, \mu_2$ satisfying conditions (3.3) to (3.8).

We set $P = C_{p,q}$ as the public matrix. However, it is not necessary to store this matrix itself, as it is fully determined by the polynomials (6.28) and q. Hence, the public key in this variant is the set $\{p, q, n, \sigma, \theta_1, \theta_2, \mu_1, \mu_2\}$, where p is given by (6.28). For the encryption procedure, we would have

$$\mathbf{c} = \mathbf{x}P + \mathbf{r}, \tag{6.29}$$

where $\mathbf{x}$ and $\mathbf{r}$ are built exactly as described in the previous chapters. However, $\mathbf{x}$ may be viewed as the coefficient vector of some polynomial $a \in \mathbb{Z}[x]$ of degree $n-1$. Hence, as observed in Proposition 2, we have

$$\mathbf{x}P = [a]C_{p,q} = [ap \bmod q]. \tag{6.30}$$

Because $\mathbf{r}$ may also be viewed as the coefficient vector of some polynomial $b \in \mathbb{Z}[x]$ of degree $n-1$, we may compute the ciphertext in terms of polynomials instead of matrices as follows:

$$c = ap + b \pmod{q}, \tag{6.31}$$

where a encodes the message, and the coefficients of b are obtained by the procedure `EPHEMERAL_KEY` described in Chapter 4.

Provided that we have the equivalence between the resultant and the characteristic matrix determinant, we are able to translate the decryption procedure described in the end of Chapter 4, in order to employ only polynomials. The ciphertext will be a polynomial c given by (6.31), and its decryption follows the steps below:

- Compute the *rounding difference polynomial* given by

$$d = \left(cg' - \mathrm{Res}(g,q)\lceil \frac{1}{\mathrm{Res}(g,q)} cg' \rfloor \right) \bmod q. \tag{6.32}$$

  Recall that g' is the polynomial such that $gg' \equiv \mathrm{Res}(g,q) \pmod q$. Also note that this polynomial is in correspondence with the modified rounding difference vector presented in (4.2).

- Consider the polynomial

$$e = \frac{1}{\mathrm{Res}(g,q)}((ah+b)g' \bmod q), \tag{6.33}$$

  which is in correspondence with the inversion error vector defined in (3.12). Applying the same procedure presented in Chapter 4, build the polynomial $\lceil e \rfloor$ such that

$$\frac{d_j}{\mathrm{Res}(g,q)} > 0 \Rightarrow \lceil e \rfloor_j = \delta \tag{6.34}$$

  and

$$\frac{d_j}{\mathrm{Res}(g,q)} < 0 \Rightarrow \lceil e \rfloor_j = -\delta. \tag{6.35}$$

- Compute the polynomial

$$\left(\lceil \frac{1}{\mathrm{Res}(g,q)} cg' \rfloor - \lceil e \rfloor \right) f' \bmod q. \tag{6.36}$$

  We claim that the factor within the parenthesis is equivalent to $af \pmod q$. In fact,

$$cg' \equiv (ap+b)g' \pmod q. \tag{6.37}$$

Hence, we have

$$cg' \equiv (a(fg + h) + b)g' \pmod{q}, \tag{6.38}$$

which yields

$$cg' \equiv af\mathrm{Res}(g, q) + (ah + b)g' \pmod{q}. \tag{6.39}$$

Thus, by (6.33),

$$cg' \equiv \mathrm{Res}(g, q)(af + e) \pmod{q}. \tag{6.40}$$

Dividing by $\mathrm{Res}(g, q)$ and rounding yields

$$\lceil \frac{1}{\mathrm{Res}(g, q)} cg' \rfloor \equiv af + \lceil e \rfloor \pmod{q}. \tag{6.41}$$

Therefore, the expression in (6.36) yields

$$aff' \pmod{q} = \mathrm{Res}(f, q)a \pmod{q}. \tag{6.42}$$

Since a is a polynomial with degree less than n, the expression above is exactly equal to $\mathrm{Res}(f, q)a$. The last step of decryption consists of dividing by $\mathrm{Res}(f, q)$ to obtain a.

This is clearly a particular case of the cryptosystem proposed in Chapter 4, with $U = C_{f,q}$, $S = C_{g,q}$ and $E = C_{h,q}$. We should make some remarks at this point. The first one is that we do not require $C_{f,q}$ to be unimodular. As a matter of fact, this would be particularly difficult, because we would have to generate a polynomial f such that $\mathrm{Res}(f, q) = \pm 1$. In our tests, we came to the conclusion that U does not *need* to be unimodular. Using a matrix with determinant different from ± 1 has the effect that attacks based on lattice reduction tend to find shorter vectors. In order to mitigate such attacks, we must choose f such that $C_{f,q}$ has small determinant (otherwise, even the short vectors of $\mathcal{L}(C_{f,q}C_{g,q})$ may be too large, which could make lattice reduction easier). We achieved this experimentally by choosing $\omega = 2$.

The other remark concerns the polynomial g, which must be chosen in such a way that $C_{g,q}$ is an $M_{\gamma,\tau}$-matrix. This can be tricky, because once the polynomial

is chosen, the coefficients of the other rows of $C_{g,q}$ can grow really fast. In order to increase the chances that $C_{g,q}$ will be in fact an $M_{\gamma,\tau}$-matrix, g must have small coefficients in absolute value and q must be sparse. We chose $q(x) = -1+q_1x+q_2x^2+\cdots+q_{n-1}x^{n-1}+x^n$, where each q_k, for $k = 1, 2, \cdots, n-1$, is -1 with probability 1% and 0 with probability 99%. The coefficients of g were selected uniformly at random from the set $\{-1, 0\}$, except for g_0, which was selected from $\{\gamma - 1, \gamma\}$.

In our experiments, we chose set 4 of parameters, described in Chapter 4, with $\omega = 2$ and $\delta = 256 + \text{rand}(256)$.

6.3 Security and Performance Analysis

All the attacks shown in Chapter 4 can be performed against this variant, and the use of structured matrices does not seem to affect the security of the scheme. However, we should make some remarks on the search space for the exhaustive search attack.

As in the previous case, the attacker only needs to try guessing two polynomials, for example f and g, and then he has to check whether $(p - fg) \bmod q$ has coefficients in the set $\mathcal{Z}_\alpha^\beta$.

For each γ, the size of the search space for g is 2^n. For h, the size of the search space is given by $(\alpha + \beta + 1)^n$. Assuming that all the coefficients are uniformly and independently chosen, the total size of the search space for the secret polynomials g and h is given by $2^n(\alpha + \beta + 1)^n$.

The search space for f depends on the bounds for its coefficients, which is given by the parameter ω. Since we choose $\omega = 2$, the search space has size 5^n

(because the coefficients will range from -2 to 2). Hence, if the attacker tries to guess f and g, for example, the size of the search space is given by 10^n.

Because of the reduction in the search space for exhaustive search, we recommend $n = 256$ as the minimum security parameter. Table 6.1 shows public key sizes (in Kbytes) for some of the recommended values of n.

Table 6.1: Public key sizes (in kB) for several dimensions.

Dimension (n)	Public Key Size (kB)
256	1.4
300	1.6
400	2.2
512	2.8

Table 6.2 shows the average time (in seconds) for key generation, encryption and decryption.

Table 6.2: Time (in seconds) for key generation, encryption and decryption.

Dimension (n)	Key generation	Encryption	Decryption
256	1.54054	0.00523	0.159355
300	2.44964	0.00702	0.400786
400	5.41614	0.013441	0.627673
512	11.5926	0.022812	0.970239

As for the hardness of exposing the secret polynomials, it is still related to the problem of lattice deformations, since this variant can be reduced to its *matrix version*, where we can replace each polynomial by the corresponding characteristic matrix.

7 CONCLUSIONS AND FINAL REMARKS

"Spins and turns, angles and curves. The shape of dreams, half remembered. Slip the surly bonds of earth and touch the face of perfection - a perfect face, perfect lace."
(Anders)

In this work, we proposed a new trapdoor function from a different kind of lattice problem, related to what we called lattice deformations. Based on that trapdoor function, we showed how to build an encryption scheme, for which an efficient variant was proposed, based on the relation between characteristic matrices and polynomials.

There are some open questions regarding the Lattice Deformation Problem. Are there triplets of subsets $\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$ for which $dLDP^{\mathcal{A},\mathcal{B},\mathcal{C}}$ is NP-complete or NP-hard? Is dLDP NP-complete or NP-hard for the triplets of subsets used to build the keys of the proposed cryptosystem?

Nevertheless, the reduction presented in this thesis has its value, as it shows that it seems to be infeasible to solve LDP in the general case (for any triplet of subsets), unless there is an efficient algorithm that always finds vectors within Minkowski's bound for any given lattice. As already mentioned, no known algorithms can always find vectors within this bound in the general case.

Attacks on the ciphertext may involve techniques other than the ones presented in this work. However, to the best of our knowledge, there seem to be no efficient methods to retrieve the message vector from the ciphertext, provided that all conditions described in Chapter 3 are fulfilled. Trying to "shorten" the pertur-

bation vector $\mathbf{r}$ turns out to be a brute force attack, since the adversary must guess the signs of the entries of $\mathbf{r}$.

The choices of parameters were essentially heuristic, being the result of an exhaustive process of trying and error. We are not aware of any other suitable set of values. The parameter γ slightly affects key size, and we could not find a set of parameters with γ less than $O(n^3)$.

As for the variant based on polynomials, the only remark is regarding the choice of the polynomial f, which corresponds to a characteristic matrix that is not unimodular. Although this does not affect the security of the scheme, it would be interesting to have a method for obtaining unimodular characteristic matrices. We are not aware of the existence of such a method.

There is also an open question as to whether it is possible to build a digital signature scheme based on lattice deformations. We could not figure out what kind of information could only be generated by the holder of the secret key, in such a way that it could be verified by applying the public key. Further research on this, and all the other open questions, is highly encouraged.

BIBLIOGRAPHY

[1] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

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

[3] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[4] Jiun-Ming Chen and Bo-Yin Yang. *A More Secure and Efficacious TTS Signature Scheme*, chapter Information Security and Cryptology - ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers, pages 320–338. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[5] Jintai Ding, Christopher Wolf, and Bo-Yin Yang. *l-Invertible Cycles for Multivariate Quadratic Public-Key Cryptography*, chapter Public Key Cryptography – PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16-20, 2007. Proceedings, pages 266–281. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[6] Tsutomu Matsumoto and Hideki Imai. *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, chapter Advances in Cryptology — EUROCRYPT '88: Workshop on the Theory and Application of Cryptographic Techniques Davos, Switzerland, May 25–27, 1988 Proceedings, pages 419–453. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[7] Lih-Chung Wang, Bo-Yin Yang, Yuh-Hua Hu, and Feipei Lai. *A "Medium-Field" Multivariate Public-Key Encryption Scheme*, chapter Topics in Cryp-

tology – CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings, pages 132–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[8] C. Dods, N. P. Smart, and M. Stam. *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings*, chapter Hash Based Digital Signature Schemes, pages 96–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[9] L. Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[10] Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 218–238, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[11] Mohssen Alabbadi and Stephen B. Wicker. A digital signature scheme based on linear error-correcting block codes. In *Proceedings of the 4th International Conference on the Theory and Applications of Cryptology: Advances in Cryptology*, ASIACRYPT '94, pages 238–248, London, UK, UK, 1995. Springer-Verlag.

[12] Robert J. McEliece. A Public-Key cryptosystem based on algebraic coding theory. Technical Report 44, Jet Propulsion Lab., CA, 1978.

[13] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, (15):19–34, 1986.

[14] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. 29th Annual ACM Symp. on Theory of Computing (STOC)*, pages 284–293, 1997.

[15] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth*

Annual ACM Symposium on Theory of Computing, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[16] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Crypto'97, Lecture Notes in Computer Science*, volume 1294, pages 112–131. 1997.

[17] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public-key cryptosystem. In *Proceedings of ANTS-III (LNCS)*, volume 1423, pages 267–288, 1998.

[18] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, November 2013.

[19] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. 37th ACM Symp. on Theory of Computing (STOC)*, pages 84–93, 2005.

[20] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Classics in Applied Mathematics. SIAM, 1987.

[21] Miklós 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*, STOC '98, pages 10–19, New York, NY, USA, 1998. ACM.

[22] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 1998.

[23] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[24] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology - Crypto'97*, volume 1294

of *Lecture Notes in Computer Science*, pages 112–131. Springer, Heidelberg, 1997.

[25] Joel Broida and S. Gill Williamson. *A Comprehensive Introduction to Linear Algebra*. Addison-Wesley, 1989.

[26] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. volume 1 of *Combinatorica*, pages 1–13. 1986.

[27] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. In *Mathematische Annalen*, volume 261, pages 515–534. 1982.

[28] C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *Math. Programming*, volume 66, pages 181–199. 1994.

[29] V. Shoup. Number Theory C++ Library (NTL) version 9.0.2. Available at `http://www.shoup.net/ntl/`, 2015.

[30] J. W. S. Cassels. *An Introduction to the Geometry of Numbers*. Springer Classics in Mathematics. Springer-Verlag, Heidelberg, 1997.

[31] J. Hoffstein, J. Pipher, and J. H. Silverman. *An Introduction to Mathematical Cryptography*. Undergraduate Texts in Mathematics. Springer, 2008.

[32] Jingguo Bi and Qi Cheng. Lower bounds of shortest vector lengths in random knapsack lattices and random NTRU lattices. *IACR Cryptology ePrint Archive*, 2011:153, 2011.

[33] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Heidelberg, 1993.