Secure Hardware Implementation of Post Quantum Cryptosystems

Mouna Nakkar

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Sciene (Information Systems Engineering) at
Concordia University
Montreal, Quebec, Canada

December 2017

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mouna Nakkar**

Entitled: **Secure Hardware Implementation of Post Quantum Cryptosystems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Sciene (Information Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Abdessamad Ben Hamza

_____ External Examiner
Dr. Khaled Galal

_____ Examiner
Dr. Jamal Bentahar

_____ Thesis Supervisor
Dr. Amr Youssef

Approved by _____
Dr. Chadi Assi, Graduate Program Director

December 6^{th},2017 _____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Secure Hardware Implementation of Post Quantum Cryptosystems

Mouna Nakkar

Concordia University, 2017

Solving a hard mathematical problem is the security basis of all current cryptographic systems. With the realization of a large scale quantum computer, hard mathematical problems such as integer factorization and discrete logarithmic problems will be easily solved with special algorithms implemented on such a computer. Indeed, only post-quantum cryptosystems which defy quantum attacks will survive in the post-quantum era. Each newly proposed post-quantum cryptosystem has to be scrutinized against all different types of attacks. Attacks can be classified into mathematical cryptanalysis and side channel attacks. In this thesis, we propose secure hardware implementations against side channel attacks for two of the most promising post-quantum algorithms: the lattice-based public key cryptosystem, NTRU, and the multivariate public key cryptosystem, Rainbow, against power analysis attacks and fault analysis attacks, respectively.

NTRUEncrypt is a family of public key cryptosystems that uses lattice-based cryptography. It has been accepted as an IEEE P1363 standard and as an X9.98 Standard. In addition to its small footprint compared to other number theory based public key systems, its resistance to quantum attacks makes it a very attractive candidate for post quantum cryptosystems. On the other hand, similar to other cryptographic schemes, unprotected hardware implementations of NTRUEncrypt are susceptible to side channel attacks such as timing and power analysis. In

this thesis, we present an FPGA implementation of NTRUEncrypt which is resistant to first order differential power analysis (DPA) attacks. Our countermeasures are implemented at the architecture level. In particular, we split the ciphertext into two randomly generated shares. This guarantees that during the first step of the decryption process, the inputs to the convolution modules, which are convoluted with the secret key polynomial, are uniformly chosen random polynomials which are freshly generated for each convolution operation and are not under the control of the attacker. The two shares are then processed in parallel without explicitly combining them until the final stage of the decryption. Furthermore, during the final stage of the decryption, we also split the used secret key polynomial into two randomly generated shares which provides theoretical resistance against the considered class of power analysis attacks. The proposed architecture is implemented using Altera Cyclone IV FPGA and simulated on Quartus II in order to compare the non-masked architecture with the masked one. For the considered set of parameters, the area overhead of the protected implementation is about 60% while the latency overhead is between 1.4% to 6.9%.

Multivariate Public Key Cryptosystems (MPKCs) are cryptographic schemes based on the difficulty of solving a set of multivariate system of nonlinear equations over a finite field. MPKCs are considered to be secure against quantum attacks. Rainbow, an MPKC signature scheme, is among the leading MPKC candidates for post quantum cryptography. In this thesis, we propose and compare two fault analysis-resistant implementations for the Rainbow signature scheme. The hardware platform for our implementations is Xilinx FPGA Virtex 7 family. Our implementation for the Rainbow signature completes in 191 cycles using a $20ns$ clock period which is an improvement over the previously reported implementations. The verification completes in 141 cycles using the same clock period. The two proposed fault analysis-resistant schemes offer different levels of protections and increase the area overhead by a factor of 33% and 9%, respectively. The first protection scheme acquires a time overhead of about 72%, but the second one does not have any time overhead.

Acknowledgments

I would like to extend my thanks and gratitude to my thesis supervisor, Professor Amr Youssef for his continuous guidance and support. He gave me an opportunity to work on cryptography research and contribute to the field, and he was always available for discussions and feedback. The research projects we conducted for this thesis gave me an in-depth knowledge and experience of the developing research in cryptography. These projects are not only an integral part for the completion of this thesis, but also a published contribution in the field of cryptography.

I would also like to thank Concordia Institute for Information Systems Engineering and all the faculty. The courses I took and the projects I conducted gave me breadth knowledge in the field of Information Security. Indeed, the course requirements for the completion of the Master's degree of Information Systems Security, MASc, significantly broadened my knowledge in the area and gave me an insight of the new research trends in the field.

Finally, I would like to thank all my colleagues in the Crytpo Laboratory, and all those who made the completion of this thesis possible.

MOUNA NAKKAR

Table of Contents

Abstract **iii**

Acknowledgments **v**

List of Figures **ix**

List of Tables **x**

List of Acronyms **xi**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Our Contribution 5

1.2.1 NTRU Countermeasure Against Power Analysis Attack 5

1.2.2 Improved Rainbow Implementation 5

1.2.3 Rainbow Countermeasure Against Fault Analysis Attacks 6

1.3 Thesis Organization 6

Chapter 2 Background and Literature Review **7**

2.1 Introduction 7

2.2 Quantum Computers and Shor's Algorithm 8

2.3 Post Quantum Algorithms 9

2.3.1 Code-based Cryptosystems 9

2.3.2 Hash-based Cryptosystems 11

2.3.3 Lattice-based Cryptosystems 13

2.3.4 Multivariate Public Key Cryptosystems 15
2.3.5 Supersingular Elliptic Curve Isogeny Cryptography 19
2.4 Side Channel Attacks 19
2.4.1 SCA methodologies 20
2.4.2 Power Analysis Attacks 21
2.4.3 Fault Analysis Attacks 21
2.5 Countermeasures Against Side Channel Attacks 22

Chapter 3 NRTU Cryptosystem **23**
3.1 Introduction 23
3.1.1 Convolution of Polynomial Rings 24
3.2 Description of the NTRUEncrypt cryptosystem 25
3.2.1 Definitions 25
3.2.2 Key Generation 26
3.2.3 Encryption 27
3.2.4 Decryption 27
3.2.5 Parameter Selection, Security Levels, and Optimization 28
3.3 Related Work 29
3.4 Proposed Countermeasures 30
3.5 Proposed Decryption Architecture 31
3.6 Results and Discussion 35

Chapter 4 Rainbow Signature Scheme **39**
4.1 Introduction 39
4.1.1 Definitions 39
4.1.2 Oil-Vinegar 39
4.2 Overview of Rainbow Scheme 40
4.3 Related Work 42
4.4 Hardware Implementation of Rainbow 43
4.4.1 Improved Architecture 44
4.4.2 Signature Process 44

4.4.3 Verification Process 45
4.4.4 Paralleled Gauss-Jordan Elimination 45
4.4.5 Affine Linear Transformation 48
4.5 Proposed Fault Analysis Countermeasures 49
4.6 Results 50

Chapter 5 Conclusions **53**
5.1 Summary and Conclusions 53
5.2 Future Work 54

Bibliography **56**

List of Figures

2.1 Code-based Public Key Encryption/Decryption 10
2.2 Merkle Hash Tree 12
2.3 A Two-Dimensional Lattice 14
2.4 Public and Private Key Construction for MPKCs Encryption/Signature 17

3.1 Top level view of the proposed masking scheme 33
3.2 Circuit for evaluating $f * e \bmod q$ 36
3.3 Circuit for evaluating the convolution multiplication $F_p * b \bmod p$ 37

4.1 Top Level Rainbow Scheme 43
4.2 Signature Block Diagram 46
4.3 Verification Block Diagram 47
4.4 Two Proposed Fault-resistant Schemes 50

List of Tables

3.2 Parameter set for NTRU [31] 28

3.3 Decryption without masking 36

3.4 Decryption with masking 37

4.1 Private Keys (Secrets) of Rainbow 45

4.2 Signature Comparison with Tang *et al.* [70] 51

4.3 Verification process Timings 51

4.4 Number of slices and clock cycles for the fault-resistant implementations . . . 52

List of Acronyms

AES	Advanced Encryption Standard
BF	Big Filed
BLISS	Bimodal Lattices Signature Scheme
CRT	Chinese Remainder Theorem
DES	Data Encryption Standard
DPA	Differential Power Analysis
DSCA	Differential Side Channel Attack
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
DLP	Discrete Logarithm Problem
DRAM	Dynamic Random Access Memory
ECC	Elliptic Curve Cryptosystem
EMA	Electromagnetic Analysis
EC-DLP	Elliptic Curve Discrete Logarithmic Problem
ET	European Telecommunications
FPGA	Field Programmable Gate Array
FLOPS	Floating Point Operations Per Second
FSM	Finite State Machine
GF	Galois Field
HFE	Hidden Field Equations

HMC	Hidden Monomial Cryptosystems
IFP	Integer Factorization Problem
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
LWE	Learning With Error
LFSR	Linear Feedback Shift Register
LUT	Look Up Table
MI	Matsumoto-Imai
MSS	Merkle Signature Scheme
MUX	Multiplexer
MPKC	Multivariate Public Key Cryptosystems
MQ	Multivariate Quadratic polynomial
NIST	National Institute of Standards & Technology
NSA	National Security Agency
NTRU	N-th Degree Truncated Polynomial Ring
NTRUEncrypt	NTRU Encryption Algorithm
NTRUSign	NTRU Digital Signature Scheme
NMR	Nuclear Magnetic Resonance (NMR)
OV	Oil-Vinegar
PQCRYTPO	Post-Quantum Cryptography Group
RAM	Random Access Memory
ROM	Read Only Memory
Ring-LWE	Ring-Learning With Error
RSA	Rivest-Shamir-Adleman cryptosystem
SAFCRYPTO	Secure Architectures of Future Emerging Cryptography
SSL	Secure Socket Layer
SCA	Side Channel Attack

SSCA	Simple Side Channel attack
SPA	Simple Power Analysis
SRAM	Static Random Access Memory
STS	Stepwise Triangular System
TLS	Transport Layer Security
UOV	Unbalanced Oil-Vinegar

Chapter 1

Introduction

1.1 Motivation

Since the revolution of computers and the Internet, governments, financial institutes, and private organizations, all shifted their services to computers with Internet based accesses. Nowadays, most of the critical data and transactions are being accessed and transmitted over the Internet or computer networks of some sort. Security of both stored and transmitted data has been a concern since the early days of computer and Internet usage. Cybersecurity is the body of technologies, processes and practices designed to protect networks, computers, programs and data from attack, damage or unauthorized access. In early days, cybersecurity was the concern of governments, financial institutes, and large organizations, but now it is the concern of everyone who uses technology. The heart of cybersecurity is encryption where it provides tools to change the form of transmitted or stored data. Cryptography is the study and practice of securely communicating between two parties despite the presence of adversaries. Cryptography and its related systems can provide several security services including privacy, confidentiality, authentication, integrity, and nonrepudiation.

There are two types of cryptographic systems available, symmetric key cryptosystems and asymmetric key cryptosystems. A symmetric crytposystem uses a key k chosen from a

space (i.e., a set) of possible keys K to encrypt a plaintext message m chosen from a space of possible messages M, and the result of the encryption process is a ciphertext c belonging to a space of possible ciphertexts C [30]. The key k is exchanged between two parties in a secure channel, and it is used for both the encryption and the decryption processes. Asymmetric key cryptography (also referred to as public key cryptography) is a class of algorithms which uses a pair key for encryption and decryption: a public key and a private key. In asymmetric key cryptosystems, we also have the same key space K, message space M, and ciphertext space C. However, the key k is really a pair of private key and public key, $k = (k_{priv}, k_{pub})$. The public key is used to encrypt the data and the private key is used to decrypt it. Public key cryptography, unlike symmetric key cryptography, allows communicating parties to establish a shared secret without a secret channel, where this fundamental difference became a very attractive feature to many communication protocols such as Internet Key Exchange (IKE) and TLS. In fact, since its invention in late 1970s, public key cryptography has been an integral part of the current communication networks. The class of asymmetric algorithms includes encryption and digital signature where the latter has proven to be one of the most important applications in cryptography. For example, the current SSL/TLS protocol relies on both public key cryptography and digital signature for authenticating websites.

The most widely used asymmetric algorithms are RSA [63], ElGamal [25], Elliptic Curve Cryptosystems (ECC) [40], and Deffie-Hellman (DH) [21]. RSA and DH, for example, are heavily used in all versions of SSL/TLS network and communication protocols. These algorithms and other current algorithms are considered to be secure, because they rely on the difficulty of solving a hard mathematical problem. In fact, the security of RSA, DH, and ECC stems from the difficulty of solving the Integer Factorization Problem (IFP), the Discrete Logarithmic Problem (DLP), and the Elliptic Curve Discrete Logarithmic Problem (EC-DLP), respectively. Indeed, a brute-force attack is always a solution for breaking any algorithm, but a large key size is almost impossible to break.

The situation is likely to change in the future, i.e. with the emergence of quantum com-

puting technology. Quantum computing, unlike digital computing, uses physical phenomenon, such as superposition and entanglement, directly to perform operations on data. The research in quantum computing is still in its infancy stage; just recently in May 2017, IBM unveiled a 17-qubit quantum computer [33]. Governments and research institutes are also supporting the development of such systems due to its high efficiency projections. In 1997, Shor [67] proposed a quantum computer algorithm that can efficiently find the prime factors of an integer N. Therefore, if a large scale quantum computer is constructed and Shor's algorithm is implemented, then almost all the current cyprtosystems such as RSA, ECC, and DH will be broken. In fact, breaking current cryptosystems will be faster than the speed of their encryption. Indeed, there is a need for secure quantum algorithms for the next era of quantum computing. In the academic world, studies of such cryptosystems and algorithms are referred to as post-quantum crytpography [6].

Even though quantum computing research is still in its infancy and a full scale quantum computing might be far off, the need for post-quantum and digital signatures standardization is exponentially rising. This is due to the fact that if a quantum computer is ever built, all current communication will be insecure which is a catastrophic scenario for the IT security. Furthermore, attackers can save the current encrypted sensitive data for such days and then break previously secured communications. Therefore, starting to secure communication with post-quantum cryptosystems now is important to prevent any information leakage to sensitive encrypted data in the future. In addition, every new technology requires time to update and upgrade all its systems around the world. We have seen and experienced attacks performed due to the existence of older outdated systems. On the other hand, the projection for the quantum computing era is a mere 15 years from now which is a short time [15]. Thus, the need for post-quantum standardization is a pressing issue to all Internet users and organizations such as National Institute of Standards and Technology (NIST), Internet Engineering Task Force (IETF), and the European Telecommunications (ET). In fact, in August 2015, NSA announced its plan to transition to quantum-computing algorithms in the near future [17].

The post-quantum research is mainly classified into lattice-based cryptography, multivariate cryptography, hash-based cryptography, code-based cryptography, and supersingular elliptic curve isogeny cryptography. The following chapter will give a brief explanation of the most popular approaches. However, our focus in this thesis is based on proposing secure hardware implementation for a lattice-based cryptography, NTRU [31,32], and a multivariate public key cryptography signature, Rainbow [23].

The attacks on cryptosystems, including post quantum cryptosystems, can be classified into three different categories: brute-force, mathematical cryptanalysis, and side channel attacks. Brute-force is when the attacker exhaustively tries all possible combinations of private keys to break the encryption. This has proven to be infeasible for large key sizes n such as 128 and 256 bits. This is due to the fact that a key with n bits can assume 2^n different combinations. Assuming the use of a supercomputer that can execute 10.51 pentaflops, i.e. 10.51×10^{15} Floating Point Operations Per Second (FLOPS), a key size of 2^{128} requires 1.02×10^{18} years to crack [2]. Indeed, this is a very long time; it is more than the age of the universe. Similarly, in mathematical cryptanalysis the attacker tries to analyze the algorithm to find a mathematical weakness to break through it. In traditional mathematical cryptanalysis, the attacker chooses the input-output pairs of the cryptographic algorithm and exploits the inner structure of the cipher to reveal the private key. Both brute-force and mathematical cryptanalysis study the theoretical weakness in the algorithm. Side channel attacks (SCAs), on the other hand, are the types of attacks that target the hardware or software implementation of the cryptosystem that inadvertently leak data. Specifically, it relies on collecting information from the physical implementation of the algorithm during execution time [42]. Collecting the timing information, power consumption, electromagnetic leaks, or acoustic information during execution time and analyzing it to break into the system are all examples of side channel attacks. Many algorithms are theoretically robust and hard to break, but their implementations can fail dramatically to side channel attacks. For example, Data Encryption Standard (DES) and Advanced Encryption Standard (AES) algorithms were broken by differential power analysis [42] and cache-timing

analysis [57], respectively. Similarly, asymmetric key cryptography can be target to side channel attacks as well. The SCA techniques applied to symmetric cryptosystems can also be applied to asymmetric public key cryptosystems such as RSA [41].

In this thesis, we propose secure hardware implementations for both NTRUEncrypt, a lattice-based cryptosystem, and Rainbow, a multivariate cryptosystem, against power analysis side channel attacks and general fault attacks, respectively.

1.2 Our Contribution

We present three major contributions. The first is a newly proposed countermeasure for NTRU encryption system against power analysis attacks. The second is an improved implementation of Rainbow signature scheme. The third contribution is a proposal of two different fault analysis attack countermeasures for Rainbow signature scheme. Some of the results presented in this thesis are published in [53].

1.2.1 NTRU Countermeasure Against Power Analysis Attack

Unprotected implementations of NTRUEncrypt are vulnerable to side channel attacks [41]. In this class of attacks, analyzing information inferred from running the algorithm on a given hardware, such as timing, power consumption, or electromagnetic emanation, can leak information about the secret key. In what follows, we present an FPGA implementation of NTRUEncrypt which is resistant to first order differential power analysis (DPA) attacks.

1.2.2 Improved Rainbow Implementation

We present an efficient high speed implementation of Rainbow signature and verification. Up to the author's knowledge, hardware implementations of the verification algorithm have not been reported in the literature before. Our signature implementation completes in 191 cycles which is an improvement over previously reported implementations [4, 10, 70] while our

verification process completes in 141 cycles. The techniques utilized in the signature implementations are the same as in the verification implementations which are based on schemes previously proposed by Bogdanov *et al.* and Tang *et al.* [10, 70]. We improve the throughput by applying high level parallelism in both solving the Gaussian elimination and performing matrix multiplications.

1.2.3 Rainbow Countermeasure Against Fault Analysis Attacks

Rainbow survived a range of attacks; however, Hashimoto *et al.* [29] presented a general fault attack that can retrieve parts of the private keys. We propose two approaches for detecting the changes in the private keys of different Rainbow layers. We compare the two proposed schemes and make recommendation to use one of the them to prevent Hashimoto *et al.* fault analysis attack.

1.3 Thesis Organization

This thesis is organized as follow. Section II provides a summary for the background information used in this thesis where some of the post-quantum cyrptosystems are briefly visited and explained. In addition, the concept of side channel attacks as well as relevant examples are presented. In Section III, we review the algorithm description of NTRUEncrypt and present a hardware implementation that resists first order power analysis attacks on the Altera Cyclone IV FPGA chip. We show that our new hardware architecture protects against first order DPA attacks. In Section IV, we move our focus to Rainbow signature scheme where we present our hardware implementation of Rainbow as well as the proposed schemes for preventing fault attack described in [29]. Our hardware implementation on the Xilinx FPGA Virtex 7 family shows an improvement over previously reported work. Finally, in Section V, we present the conclusion and future work.

Chapter 2

Background and Literature Review

2.1 Introduction

This section introduces relevant mathematical and background information for the secure hardware implementation of the post-quantum cryptosystems proposed in this thesis: NTRUEncrypt and Rainbow signature scheme. In the first part of this section, we briefly review research on post-quantum cryptosystems. In the second part, we present the currently used methods to analyze the security of post-quantum cryptosystems which include mathematical cryptanalysis and side channel attacks.

Perhaps one of the most important concepts to consider in building or proposing a new cryptosystem is time complexity. It may not be obvious or relevant at first sight, but time complexity which is generally referred to as running time is an important security parameter for any algorithm. Expressed by *Big O notation*, running time is the amount of time taken for an algorithm to run as a function of the length of the string representing the input. It is estimated by counting the number of operations performed by an algorithm [68, 69]. Generally, complexity can be classified as constant, linear, logarithmic, polynomial, and exponential. Perhaps, the most promptly used terms are the last two.

An algorithm is said to be executed in polynomial time if the number of steps required

to solve the algorithm for a given input is $O(n^k)$ where k is a non-negative integer and n is the complexity of the input. The algorithms in this category are accepted to be efficient. On the other hand, an algorithm is said to be solvable in exponential time if $T(n)$ is bounded by $O(2^{n^k})$ for some constant k; an example for an equation of exponential time is 2^n.

2.2 Quantum Computers and Shor's Algorithm

In 1997, Shor [67] proposed a quantum computing algorithm for solving the integer factorization problem and the discrete logarithm problem in polynomial time. The algorithm was a breakthrough and a motivator at the same time for both the quantum computing community and the cryptography community. On one hand, researchers were motivated to accelerate the building of a quantum computer. In 2017, IBM unveiled a 17-qubit quantum computer through the IBM Quantum Computing project [33]. On the other hand, other researches focused on demonstrating Shor's algorithm on the available quantum computer. For example, in 2001, Lieven *et al.* [72] were able to factor a small number, 15, into 3 and 5 on the Nuclear Magnetic Resonance (NMR) quantum computer implementation of 7-qubit. Later in 2012 [48], a larger number, 21, was factored using qubit recycling. More importantly, the security community realized that breaking the current cryptosystems such as RSA, ElGamel, DH, and ECC is inevitable in the quantum computing era. This realization led to the active developments of post-quantum cryptosystems which focus on resisting quantum attacks; i.e. Shor's algorithm executing on a large scale quantum computer. The post-quantum research development is now heavily endorsed and supported by governments, institutes, and organizations such as the European Commission and the Japanese Society for the Promotion of Science to projects such as the SAFECRYPTO [64], PQCRYPTO [6], CryptoMathCrest [20], and others.

2.3 Post Quantum Algorithms

Post quantum cryptosystems can be categorized as lattice-based cryptosystems, multivariate cryptosystems, hash-based crytpsystems, code based cryptosystems and isogenous supersingular elliptic curves-based cryptosystems [6,22]. The most promising post-quantum algorithms include lattice-based cryptosystems, multivariate cryptosystems, code-based cryptosystems, and hash-based cryptosystems. Thus, in this section we briefly look at the most promising ones and give examples of each one of them.

2.3.1 Code-based Cryptosystems

This type of cryptography is related to algorithms that rely on error-correcting codes. The first code-based cryptosystem was the McEliece algorithm developed in 1978 [50] using random Goppa Code [5]. Though it did not gain acceptance in the cryptography community at that time because of its large key size, it is now surfacing as a good candidate for quantum computing era because of its resistance to Shor's algorithm. In 2015, McEliece's algorithm was recommend by the Post-Quantum Cryptography (PQCRYPTO) Group sponsored by the European Commission as a candidate for long term protection against quantum attacks.

The security of the McEliece's algorithm is based on the hardness of decoding a general linear code. Similar to any asymmetric cryptosystem, McEliece's algorithm has a private key and public key where the private key is generated from an error-correcting code with the ability of correcting t errors. The public key is selected from the private key by disguising the selected code as a general linear code. Figure 2.1 shows the encryption and decryption processes of McEliece's original code-based public key cryptosystem. The plaintext goes through a linear expansion where a generator matrix representing the public key allows everyone to encrypt. The encryption process uses a carefully selected binary Goppa code word with randomly chosen errors to produce a ciphertext. The decryption process is the reverse of the encryption process where only a legitimate user who knows the trapdoor can remove the errors and recover the

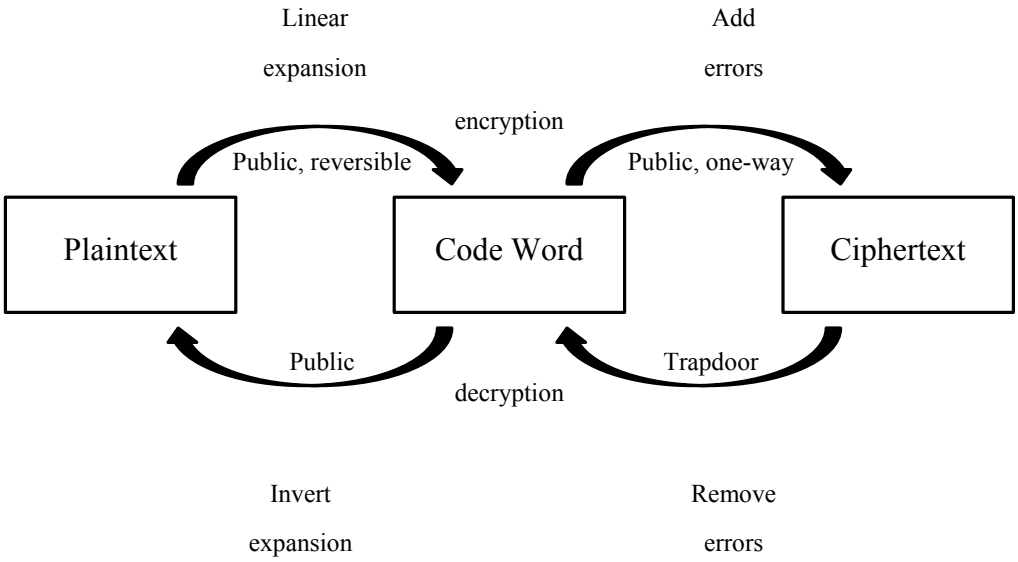


Figure 2.1: Code-based Public Key Encryption/Decryption

original text. The trapdoor in this case allows the legitimate receiver to have a polynomial time decoding algorithm which enables the removal of previously added noise and recovery of the plaintext. The security basis of McEliece revolves around two facts: recovering the decoding polynomial is considered a hard problem and the public key generator matrix is hardly distinguishable from any random matrix [65].

Niederreiter cryptosystem [56] is a spin-off from the McEliece cryptosystem where both are considered to have the same level of security [65]. The advantages of using Niederreiter scheme is efficiency where its encryption is much faster than McEliece's algorithm [65]. In addition, it can be used to construct a digital signature and zero-knowledge authentication protocol. However, the signature scheme is complex, uses large public keys, and not scale-

able [65]. Though the security of McEliece's algorithm is well understood where it resisted several cryptoanalysis for a long time, it is considered impractical. This is due to its complexity and its large public key size that is in the order of megabytes. The digital signature scheme is also considered impractical. However, researchers are constantly proposing enhancements and modifications to the original McEliece's and Niederreiter's to address issues such as complexity and efficiency. In 2008, Berstien *et al.* published an attack as well as a fix to prevent this attack [8] where they showed that security of the original proposed algorithm can be broken in $1,400$ days with a single $2.4GHz$ Core 2 Quad CPU. Thus, they proposed new parameters for both McEliece and Niederreiter's schemes for added security. Other attacks have failed and the McEliece code-based cryptosystem is still considered a strong candidate for post-quantum cryptography.

2.3.2 Hash-based Cryptosystems

From its name, hash-based cryptography is based on the security of hash functions, and thus far it is limited only to the digital signature schemes. The first hash-based signature scheme is the Merkle Signature Scheme (MSS) proposed in the late 1970s by Merkle [51, 52]. It has been developed significantly since then, but the original idea is based on one-time signature schemes and hash-trees. Specifically, many one-time signature key pairs are combined into a tree like structure. The hash-tree is a hierarchical data structure that is composed of leaf nodes. Every leaf node is associated with data block, and the non-leaf nodes are the hashes of the leaf node as depicted in Figure 2.2. The hash values are all concatenated to form a tree structure where the final signature is the repeated hashes of the sub-nodes. A one-time signature scheme is a stand-alone miniature digital signature where it can be used only once for a given key pair. Its security relies on the security of the hash function it is using. This one-time signature scheme is used as the template of hash-based schemes, and the structure of hash-based schemes remains the same with the varying hash functions. Merkle's original scheme uses a large number of one-time signature key pairs. This is because the use of only one-time signature scheme requires

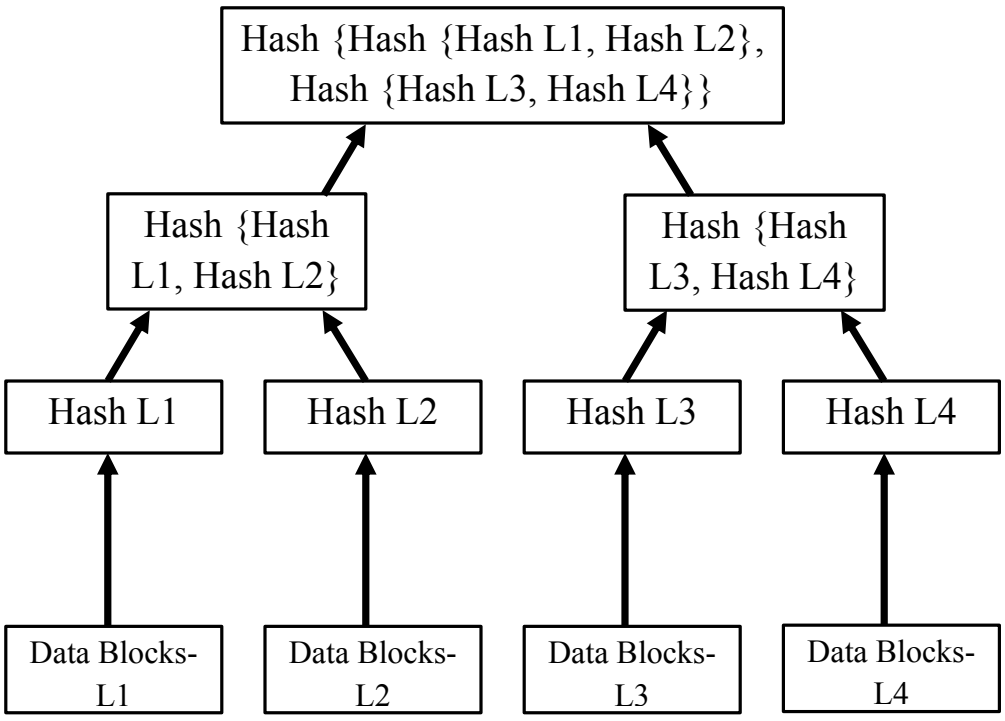


Figure 2.2: Merkle Hash Tree

a new key pair for each new signature which is impractical. Thus, Merkle combines a large number of one-time signature key pairs into one single structure where the public and private keys are constructed from these many key pairs.

Merkle's signature has been accepted by the community to be a good alternative to the traditional signature schemes such as RSA Signature and Digital Signature Algorithm (DSA). The security advantage of the Merkle's Signature is its resistance to quantum attacks, but the scheme depends highly on the security of the hash function it uses.

Lamprot's signature is another hash-based cryptosystem that combines a one-time signature with Merkle's hash-tree structure for multiple and more efficient signing scheme [43]. Lamport's signature is built from the cryptographic one-way hash function. Similar to the Merkle's

signature, Lamport's signature is believed to withstand quantum attacks. A more recent hash-based cryptosystems are the XMSS [14] and SPHINCS [7] schemes.

2.3.3 Lattice-based Cryptosystems

Lattice-based cryptosystem is the generic term for cyrptosystems that involves lattices in its construction. Algorithms belong to the lattice-based cryptosystems have proved to be good candidates for both classical and post-quantum computer era. The security of such systems stems from the assumption that many lattice-based construction are based on certain well-studied computational lattice problems that cannot be solved efficiently, even on a quantum computer. In addition, the hardware implementations of lattice-based systems tend to be faster and more efficient than other post-quantum cryptosystems. Types of this cryptosystems include Learning With Error (LWE) [61], Ring-Learning With Error (Ring-LWE) [47], NTRUEncrypt [31, 32], NTRUSign [54, 55], homomorphic encryption schemes [12, 26], and Bimodal Lattices Signature Scheme (BLISS) [24]. Among all the lattice-based cryptographic systems, NTRUEncrypt has proved to be the most practical and efficient to implement. In addition, its security has been studied for years without any feasible attack. To appreciate the security primitives of NTRUEncrypt, the following subsections will look into lattices as well as their hard problems.

A *vector space*, V is a subset of $\mathbb{R}^m$ with the property that

$$\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} \in V \; for \; all \; \boldsymbol{v_1}, \boldsymbol{v_2} \in V \; and \; all \; \alpha_1, \alpha_2 \in \mathbb{R} \tag{2.1}$$

Equivalently, a vector space is a subset of $\mathbb{R}^m$ that is closed under addition and under scalar multiplication by elements of $\mathbb{R}$ [31]. A lattice is similar in definition to a vector space except that the multiplication is restricted to an integer. A lattice L is a discrete subset of the real vector space $\mathbb{R}^m$. Every lattice in $\mathbb{R}^m$ can be generated from a basis for the vector space by forming all linear combinations with integer coefficients. One can imagine an Euclidean space, where a

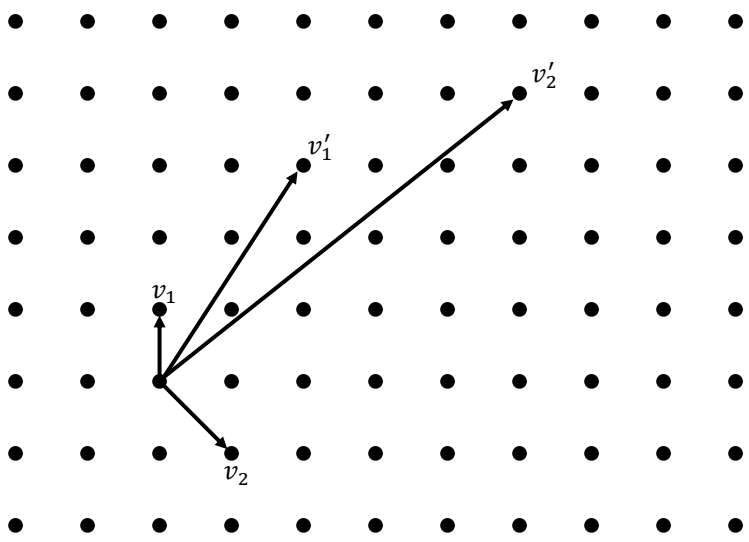


Figure 2.3: A Two-Dimensional Lattice

random set of linearly independent vectors is chosen, and the lattice consists of all points that are integer linear combinations of these vectors.

Formally [30], let $v_1, \cdots, v_n \in \mathbb{R}^m$ be a set of linearly independent vectors. The *lattice* L generated by $v_1, \cdots, v_n$ is the set of linear combinations of $v_1, \cdots, v_n$ with coefficients in $\mathbb{Z}$,

$$L = \{a_1 v_1 + a_2 v_2 + \cdots + a_n v_n : a_1, a_2, \cdots, a_n \in \mathbb{Z}\} \tag{2.2}$$

A *basis* for L is any set of independent vectors that generates L where any two sets have the same number of elements. The dimension of L is the number of vectors in a basis for L. A lattice has many bases. Two possible bases, $\{v_1, v_2\}$ and $\{v_1', v_2'\}$ for a two dimensional lattice are shown in Figure 2.3. A lattice is integral if it is contained in n-dimensional integer vector space $\mathbb{Z}^n$ and it is called rational if it is contained in $\mathbb{Q}^n$, where $\mathbb{Q}$ denote the set of rational numbers . $\mathbb{Z}^n$ is a simple example of a lattice in $\mathbb{R}^n$.

Hard Problems in Lattices

The two fundamental hard problems in lattice-based cryptography are the Shortest Vector Problems (SVP) and the Closest Vector Problems (CVP). The latter problem is finding a lattice point closest to the specified target point with respect to a lattice basis. SVP, on the other hand, is finding the shortest vector in a lattice given an arbitrary basis with very long vectors in very large dimensions. Both of these problems are considered hard mathematical problems which are the security grounds of lattice-based cryptography.

Hoffstien *et al.* stated that [30], given a vector $w \in \mathbb{R}^m$ that is not in L, CVP problem is to find a vector $v \in L$ that is closest to w, i.e., to find a vector $v \in L$ that minimizes the Euclidean norm $\|w - v\|$. NTRUEncrypt is based on the CVP where the encryption is the task of selecting a target point and decryption is the task of mapping this target point back to the closet lattice point.

2.3.4 Multivariate Public Key Cryptosystems

The term Multivariate Public Key Cryptosystems is the generic term for cryptographic systems based on the primitive of multivariate polynomials over a finite field, K. Algorithms of this type proved to be very good candidates for the post-quantum era. Though this type of cryptosystem includes both asymmetric public encryption systems and signature schemes, it is more successful with the latter due to the output size of the signature. Primarily, the multivariate signature schemes provide the shortest signature among other post-quantum signature schemes. MPKC schemes were first introduced by Matsumoto *et al.* [49] and Tsujii *et al.* [71] in the mid 1980s. Although both these systems were broken by Patarin *et al.* [59] and Hasegawa *et al.*, respectively, the idea was the foundation of this class of cryptography. Later on, several systems were developed such as the Hidden Monomial Cryptosystems (HMC) and Hidden Field Equations (HFE) [60], and Rainbow [23].

Construction of MPKCs

The security aspect of MPKCs stems from the difficulty of solving a set of multivariate equations over a finite field, K. If the set of polynomials has a degree of two, the system is referred to as multivariate quadratics. For efficiency reasons, most of the described MPKCs systems are quadratic polynomials in several variables over a small finite field, K with q equations as shown in the following set of equations.

$$\begin{aligned} p^{(1)}(x_1, \cdots, x_n) &= \textstyle\sum_{i=1}^{n} p_{ij}^{(1)} x_i \cdot x_j + \sum_{i=1}^{n} p_i^{(1)} x_i + p_0^{(1)}, \\ p^{(2)}(x_1, \cdots, x_n) &= \textstyle\sum_{i=1}^{n} p_{ij}^{(2)} x_i \cdot x_j + \sum_{i=1}^{n} p_i^{(2)} x_i + p_0^{(2)}, \\ &\cdots, \\ p^{(m)}(x_1, \cdots, x_n) &= \textstyle\sum_{i=1}^{n} p_{ij}^{(m)} x_i \cdot x_j + \sum_{i=1}^{n} p_i^{(m)} x_i + p_0^{(m)}. \end{aligned} \tag{2.3}$$

Given the m quadratic polynomials $p^{(1)}(x), \cdots, p^{(m)}(x)$ as shown in Equation 2.3 in the n variables $x_1, \cdots, x_n$, the multivariate quadratic polynomial (MQ) problem is the task of finding a vector $\hat{x} = (\hat{x}_1, \cdots, \hat{x}_n)$ such that $p^{(1)}(\hat{x}) = \cdots = p^{(m)}(\hat{x}) = 0$. This task is considered the hard problem of the MPKCs, and thus this is the security basis of such systems.

The public key in MPKCs is given by simple quadratic map $F : k^n \rightarrow k^m$ called the central map which represents a map from k^n field to k^m field. To hide the mapping structure in the central map, F is placed between two linear transformations, L_1 and L_2 in the k^n and k^m fields, respectively. The final public key is the composed map of $P = L_1 \circ F \circ L_2$. The private key, on the other hand, consists of the three maps L_1, F, and L_2. Incidentally, this process is standard for both the encryption and the signature scheme as shown in Figure 2.4. To encrypt a message $z \in k^n$, the public key is evaluated through the mapping process P and generates a ciphertext $w = P(z) \in k^m$. To decrypt the generated ciphertext, $w \in k^n$, the mapping process is reversed where the message can be retrieved by the map $z = L_2^{-1} \circ F^{-1} \circ L_1^{-1}$. Here, evaluating $F^{-1}(x)$ requires finding a solution of the polynomials under the central map. Usually, this is equivalent to finding the solution of the set of polynomials by using Gaussian elimination process as shown in Chapter 4 for Rainbow signature. For the encryption scheme,

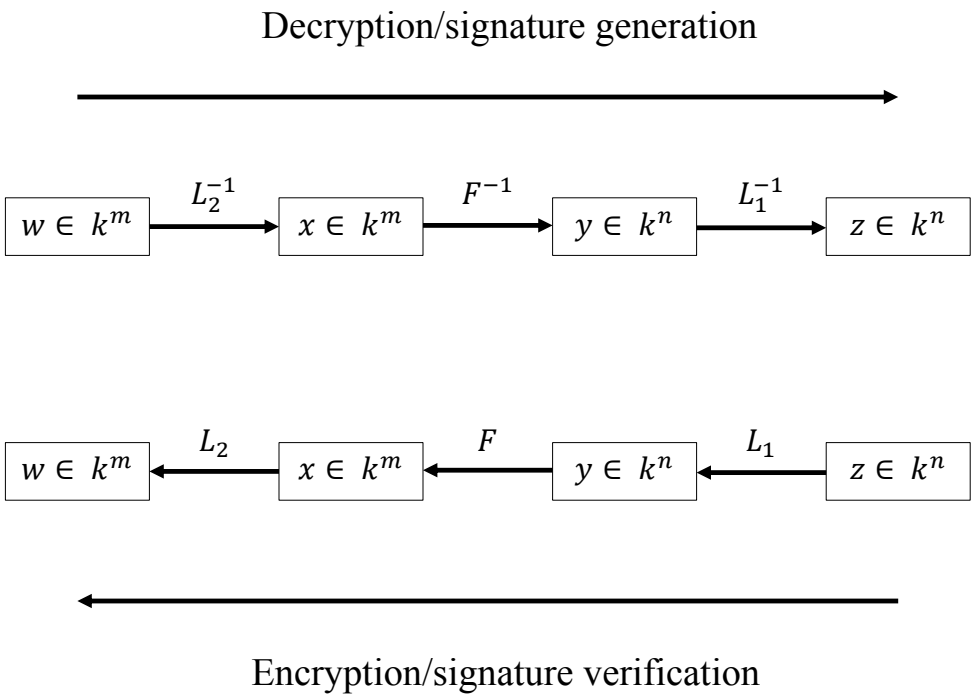


Figure 2.4: Public and Private Key Construction for MPKCs Encryption/Signature

m is chosen to be greater than or equal to n, $m \geq n$, to ensure that the decryption process outputs a unique plaintext, $z \in k^n$ [22].

The signature scheme is similar in principle to the encryption scheme. They both have the same central map; however, the parameters m and n are chosen such that $m \leq n$ to ensure that one can sign any document [22]. To generate a signature, the message, d, is hashed through a secure hash function such that $w = H(d) \in k^m$. The hash is then fed into the reverse mapping process as shown in Figure 2.4. Similar to the encryption process, the signed message is $z \in k^n$. To verify the signature and check for authenticity, the hash of the message, d, is computed to get $w = H(d) \in k^m$, then the public key is used to compute $w' = P(z) \in k^m$. If $w' = w$ holds, then the signature is valid and thus accepted; otherwise, it is rejected.

Classifications of MPKCs

MPKCs algorithms are classified into four different classes according to their central map construction: Big Filed (BF) Matsumoto-Imai Scheme (MI) [49], Hidden Field Equations (HFE) [60], Stepwise Triangular Systems (STS) [66, 71], and Unbalanced Oil and Vinegar systems (UOV) [23, 38]. In the Big Field type, the central map, F, is given by the extension of the original field, k. Let N be a divisor of $gcd(n, m)$ and K an algebraic extension of k with $[K; k] = N$. The central map F is given by $F = \psi \circ F \circ \phi$, where $\phi : k^n \rightarrow K^{n/N}$ and $\psi : K^{m/N} \rightarrow k^m$ are the one-to-one maps and $F : K^{n/N} \rightarrow K^{m/N}$ is a map given by polynomials over K as shown in Equation 2.4.

$$F : k^n \xrightarrow{\phi} K^{n/N} \xrightarrow{F} K^{m/N} \xrightarrow{\psi} k^m \tag{2.4}$$

The Big Field scheme was first developed by Matsumoto-Imai in the late 1980s [49], and it was considered a breakthrough in the filed of MPKCs. MI was broken by Patarin *et al.* [59]; however, he later developed an improvement of the original MI scheme and called it Hidden Field Equation (HFE) where the mapping function F is computed by the Berlekamp algorithm. Kipnis-Shamir [39] developed an attack to recover the two linear transformation keys by the MinRank attacks [28]. The STS types, on the other hand, were developed independently for both encryption and signature schemes by Tsujii [71] and Shamir [66]. For these two types, attacks to recover part of the key were proposed by [19] and others. The Unbalanced Oil-Vinegar (UOV) signature scheme [38] from the Oil-Vinegar (OV) family has sustained attacks for 15 to 20 years. The Oil-Vinegar family can be classified as balanced Oil-Vinegar, unbalanced Oil-Vinegar, and Rainbow. In the next chapter, a detailed explanation of the Rainbow signature scheme is presented.

2.3.5 Supersingular Elliptic Curve Isogeny Cryptography

This type of cryptography is considered fairly new; it was introduced only 10 years ago. Although it has not been standardized yet, it is considered a good candidate for post-quantum computing era. This type is now being proposed in many crytpographic protocols, signature schemes, and key exchange methods [34, 44]. It is based on replacing the widely used Diffie-Hellman and elliptic curve Diffie-Hellman key exchange methods with one that uses properties of supersingular elliptic curves to resist quantum attacks. It also claims to offer forward secrecy which is an important property for securing communication protocols. Forward secrecy protects past session keys from being compromised if the long term key got compromised.

2.4 Side Channel Attacks

The security analysis for each proposed post-quantum algorithm is critical to its survival. Each proposed algorithm is studied and analyzed for weaknesses and vulnerabilities. Unlike mathematical cryptanalysis, SCAs are the types of attacks concerned with finding vulnerabilities in the implementation of the algorithm rather than finding theoretical weaknesses in the algorithm itself. Insecure hardware implementation of cryptographic algorithms has proven to be a critical issue and a security threat. If algorithms are not implemented carefully, they can be susceptible to SCAs. This is because SCA specifically analyzes the physical leaked information such as sound, power consumed, electromagnetic, and other physical phenomenon occurring during run-time. Strong algorithms such as AES and DES have been broken by SCA [16, 41, 42]. With the deployment of cryptographic algorithms within embedded devices, SCAs are becoming essential part of the security measure for the implementation of any algorithm. Cryptography is no longer limited to communication protocols; instead, cryptography is now widely used in TV set-top boxes, cellar phones, bank cards, and prepaid cards which are easily accessible to attackers. Insecure implementations can be as damaging as weak cryptosystems.

2.4.1 SCA methodologies

In order to carry out a side channel attack, the attacker must be able to control the computation process, access the crytographic module, and analyze the data. In turn, there are two ways to control the computation process in SCA where the attacker can be active or passive. In passive attacks, the attacker does not interfere with the computation process. The attacker is assumed to be able to collect the needed information from the module without interfering in its behavior. In active attacks, on the other hand, it is assumed that the attacker can affect the behavior of the targeted module.

The task of accessing the module can be classified into three categories invasive, semi-invasive, and non-invasive [1]. Invasive attacks require a direct access to the internals of the device by using proper probing equipment. This will allow the attacker to monitor data buses or internal registers. In contrast, with the semi-invasive attacks, the attacker does not need to have a direct access to internal nodes; instead, the attacker changes the contents of internal nodes such as memory and or internal registers by inducing a fault using external equipment such as ionized laser beam. The adversary then monitors the output based on this change and collects its output to recover the secret key or parts of it. Non-invasive attacks require no interference of the physical device at all. In these types of attacks, the adversary takes advantage of the unintentionally leaked information during execution time. An example for this non-invasive is the timing attack where the adversary takes advantage of the time taken to execute the algorithm to retrieve information about the secret key.

There are two ways to analyze the sampled data to fully recover the private key or parts of it, and they are Simple Side Channel attack (SSCA) and Differential Side Channel Attack (DSCA). In SSCA, the output is analyzed directly according to a single trace of the leaked data in order to retrieve the full private key or parts of it. In DSCA, on the other hand, the attacker collects numerous traces of leaked data and statistically analyze them. The attacker exploits the correlation between these collected traces to retrieve the full private key or parts of it.

There are several types of implementation dependent attacks considered in literature as

mentioned above. SCAs are based on analyzing physical properties such as time, time-cached, electromagnetic properties, power consumption, acoustic effects, and others. In this thesis, we focus on Differential Power Analysis (DPA) attacks for NRTUEncrypt system. We also focus on fault analysis attacks for Rainbow signature scheme.

2.4.2 Power Analysis Attacks

Power analysis attacks [41] are categorized into Simple Power Analysis (SPA), Differential Power Analysis (DPA) which can also be sub-categorized into first order and higher order DPA, and Correlation Power Analysis (CPA) [13]. SPA leaks information about operations being executed by observing a single power trace. On the other hand, DPA and CPA require the collection of a large number of traces using the same key for statistical analysis. CPA uses similar techniques to DPA, but with more refined statistical analysis. With DPA, after collecting a large number of power traces with the same key, the attacker makes a guess on bits of the secret key. Knowing the ciphertext and the algorithm, the attacker computes values of internal registers pertaining to this guess. The power traces are then partitioned based on the computed values and the guess is accepted if there is a significant statistical difference between partitioned traces and original ones.

2.4.3 Fault Analysis Attacks

Fault analysis attacks are physical attacks where the attacker is assumed to be able to inject faults in the targeted device and analyze the outcome of the faulty computations in order to retrieve partial or full information about the private key. The faults in this class of attacks are injected by subjecting the device to unexpected environmental conditions such as high temperature, voltage surge above the operating range, induced magnetic field, excessive clocking, or ionized radiation. The induced conditions are assumed to influence the processed data and corrupt the output. This incorrect results and data corruption may allow the attacker to gain some information about the private key or gain information about internal status of the device. Fault

analysis attacks were first introduced by Boneh *et al.* [11] where they showed that extracting part of the RSA decryption/signing keys can be easily done by inducing errors in the Chinese Remainder Theorem (CRT) implementation of the RSA algorithm and observing its faulty outputs. Other research followed the same concept to break cryptosystems such as NTRUEncrypt and others (e.g., see [35], [18], [36]).

2.5 Countermeasures Against Side Channel Attacks

Perhaps the best countermeasures against side channel attacks is building secure hardware devices especially for the SCAs that rely on hardware implementations such as differential power analysis and fault analysis. For these two types, a secure hardware architecture is sufficient for protecting the hardware and not leaking information during execution time. In this thesis, we propose a secure hardware implementation for NRTUEncrypt against DPA where we split the ciphertext into two shares and process them in parallel. The internal processed data are always coupled with randomly generated polynomials which makes it hard to trace the data or internal states of the registers. This is because the consumed power is paired with randomly freshly generated shares in each decryption. For the Rainbow signature, we propose two solutions to detect changes in the private keys. If any change is detected, the signature process is halted.

Chapter 3

NRTU Cryptosystem

3.1 Introduction

Since its introduction in 1996, NTRUEncrypt (also known as NTRU), has proven to be a very efficient public key cryptosystem. NTRUEncrypt is an asymmetric public key crytposystem, and it provides faster and lighter hardware implementations compared to the traditional number theory based public key cryptosystems such as Diffie-Hellman, RSA, and elliptic curve based cryptosystems. The efficiency of NTRUEncrypt is due to the fact that the most complex operation in NTRU is the polynomial multiplication which is faster than the other cryptosystems such as RSA, Deffie-Hellman, ElGamal, and elliptic curve. On the other hand, the security of NTRU is based on the difficulty of finding the shortest vector problem in a lattice and stems from the interaction of polynomial systems with the independence of the reduction modulus of two numbers p and q [31, 32]. The encryption process is based on polynomial ring arithmetic modulo two primes (integer or polynomial) while the decryption process is the reverse of the encryption process using an elementary probability theory with a chance of failure. In addition to its small footprint compared to other number theory based public key systems, its resistance to quantum attacks based on Shor's algorithm makes it a very attractive candidate for post quantum cryptosystems.

3.1.1 Convolution of Polynomial Rings

Rings are algebraic objects that have two operations, addition and multiplication, which are connected through the distributive law. NTRUEncrypt is based on the convolution polynomial rings with degree of $N-1$ and integer coefficients. The basic two operations for NTRUEncrypt are addition and multiplication. Addition is a straightforward task of adding two polynomials where the coefficients of a similar degree are added together as shown in Equation 3.1. In the addition operation, the resultant polynomial will remain within $N-1$ degree. However, the multiplication operation over a polynomial rings is different. The number of coefficients will be doubled because new terms, with degree higher than N-1, are created. The modulo operation on this intermediate result will reduce all X^N terms to 1 respecting the $X^N \equiv 1$ modulo rule. Higher degree terms will reduce; for example, X^{N+1} will reduce to X, X^{N+2} will reduce to X^2, and so forth. As for the coefficients, the general rule is that the k^{th} coefficient c_k is the dot product of the coefficients a and b where the coefficients of b are reversed and rotated $k+1$ coefficients [31] as shown in the following set of equations.

$$
\begin{aligned}
a + b &= (a_0 + b_0) + (a_1 + b_1)X + \cdots + (a_{N-1} + b_{N-1})X^{N-1} \\
a * b &= c_0 + c_1 X + c_2 X^2 + \cdots + c_{N-2} X^{N-2} + c_{N-1} X^{N-1} \\
c_k &= a_0 b_k + a_1 b_{k-1} + \cdots + a_k b_0 + a_{k+1} b_{N-1} + a_{k+2} b_{N-2} + \cdots + a_{N-1} b_{k+1}
\end{aligned}
\tag{3.1}
$$

The multiplication over polynomial ring is referred to NTRU cyclic convolution over polynomial ring and is denoted by $\mathbb{Z}[X]/X^N - 1$ where $\mathbb{Z}[X]$ represents a polynomial ring with integer coefficients and the division of $X^N - 1$ represents the reduction of the polynomial.

3.2 Description of the NTRUEncrypt cryptosystem

3.2.1 Definitions

The NTRU encryption algorithm is parameterized by the following parameters.

- N The degree parameter. N is prime, and it defines the degree of polynomials used in the polynomial convolution ring, R.
- q Large modulo. It is mostly used to reduce the convolution operation in the polynomial ring.
- p Small modulo. It is used to reduce the random generation components in the encryption process to reduce message space.
- d_f An integer value fixes the form of the private key polynomial, f. The number of positive ones in f is d_f, while the number of negative ones is $d_f - 1$.
- d_g An integer value fixes the form of the public key polynomial, g. The number of positive ones in g is d_g, also the number of negative ones is d_g.
- d_r An integer value fixes the form of the random polynomial, r. The number of positive ones in r is d_r, also the number of negative ones is d_r. The random polynomial, r, is used in the encryption process.
- d_m An integer value fixes the form of the message polynomial, m, to be encrypted.

The following properties are important for NTRUEncrypt cryptosystem:

- the parameters (N, p, q) are public,
- p and q are relatively prime, gcd(p, q) = 1, and
- q is much larger than p, $p << q$.

In this thesis, we focus on the case where q is in the form of 2^n because this allows us to perform the modular operations efficiently by simply truncating the results to n bits. Furthermore, we also restrict our focus to the case $p = 3$ since it allows more efficient implementation of the convolution circuits in both the decryption and encryption algorithms.

Let R, R_p, and R_q be the polynomial rings

$$R = \frac{\mathbb{Z}[x]}{x^N - 1}, R_p = \frac{\mathbb{Z}/p\mathbb{Z}[x]}{x^N - 1}, R_q = \frac{\mathbb{Z}/q\mathbb{Z}[x]}{x^N - 1}.$$

The product of two polynomials $a(x), b(x) \in R$ is given by

$$a(x) * b(x) = c(x)$$

where

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{N-1} a_i b_{N+k-i} = \sum_{j=k-i \pmod N} a_i b_j. \tag{3.2}$$

For any positive integers d_1 and d_2, let $\tau(d_1, d_2)$ denote the set of ternary polynomials given by

$$\left\{ a(x) \in R : \begin{array}{l} \text{a(x) has } d_1 \text{ coefficients equal to 1,} \\ \text{a(x) has } d_2 \text{ coefficients equal to -1,} \\ \text{all other coefficients equal to 0} \end{array} \right\}$$

In what follows, we summarize the three main operations in the NTRUEncrypt cryptosystem [31, 32].

3.2.2 Key Generation

For the public key, the user:

- selects a private $f(x) \in \tau(d_f, d_f - 1)$ that is invertable in R_q and R_p,

- selects a random polynomial $g(x) \in \tau(d_g, d_g)$,
- computes the inverse of $f_q(x) = f^{-1}(x)$ in R_q and $F_p(x) = f^{-1}(x)$ in R_p, and
- evaluates $h(x) = f_q(x) * g(x)$ in R_q.

The polynomial $h(x)$ is the user's public key and the corresponding private key is the pair $(f(x), F_p(x))$.

3.2.3 Encryption

The encryption for plaintext $m(x) \in R_p$ is performed as follows. The user:

- selects an ephemeral key $r(x) \in \tau(d_r, d_r)$, and
- evaluates the ciphertext $e(x) = pr(x) * h(x) + m(x) \bmod q$.

The encrypted message is the encrypted message $e(x)$.

3.2.4 Decryption

The decryption procedure requires the following three steps. The user:

- evaluates $a(x) = f(x) * e(x) \bmod q$,
- evaluates $b(x)$ = Centerlift$(a(x))$ such that its coefficients lie in the interval $(-q/2, q/2]$, and
- evaluates $m = F_p(x) * b(x) \bmod p$.

The last step of the decryption process requires the user to compute the inverse of the polynomial F_p of the secret $f \mod p$. The result of the decryption process will recover the original message m.

Table 3.2: Parameter set for NTRU [31]

Security Level	N	p	q	$\mathbf{d_f}$	$\mathbf{d_g}$	$\mathbf{d_r}$
Moderate security	167	3	128	61	20	18
High security	263	3	128	50	24	16
Highest security	503	3	256	216	72	55

3.2.5 Parameter Selection, Security Levels, and Optimization

The parameter selection of (N, p, q) and d_f, d_g, d_r defines the different security levels of NTRUEcrypt cryptosystem. The (N, p, q) parameters are public, while d_f, d_g, and d_r define different spaces. The set of spaces for defined d_f, d_g, and d_r will be as follows

$$L_f(d_f, d_f - 1), \quad L_g(d_g, d_g), \quad L_r(d_r, d_r)$$

where $f \in L_f$, $g \in L_g$, and $r \in L_r$. It is important to select p and q not to have a common divisor, i.e. $gcd(p, q) = 1$, in order to compute the inverse of a certain polynomial. The parameter selection has been changed over the years and is susceptible to change in the future. However, for our project, we chose parameters similar to those suggested in the original NTRU proposal [31] as shown in Table 3.2. Typical parameter sets that yield security levels similar to 1024-bit RSA and 4096-bit RSA are $(N, p, q) = (251, 3, 128)$ and $(N, p, q) = (503, 3, 256)$, respectively.

One of the objectives of developing NTRUEncrypt cryptosystem is high speed and low memory requirements. Hoffistein *et al.* [31] proposed NTRU as an efficient replacement of the most popular encryption algorithm RSA. In addition to its inherent high speed algorithm, there are efficient implementation practices that can be taken to achieve optimal performance. For example, during the last step of the decryption process, we multiply the inverse of the secret key, $F_p(x)$ with the intermediate polynomial $b(x)$ in order to obtain the original message, m, i.e. $m = F_p(x) * b(x)$. In this step, if we choose the private key of the form of $f = 1 + p * F$, where F is a random polynomial with dF non-zero coefficients, we can accelerate performance. This is

because obtaining the inverse of this form of polynomial, $F_p = f^{-1} \mod p$ eliminates the final convolution multiplication in the final step of the decryption process thus increases performance. In addition, this selection will save computational energy, circuits area, and memory storage. Additional practices can be implemented to gain high speed, but are not listed in this thesis.

3.3 Related Work

The literature is rich with countermeasures against DPA attacks including solutions at the algorithm level, circuit level, and gate level. Example of solutions at the algorithm level is masking which makes the secret independent of processed data [9, 58]. Examples of gate level approaches are those solutions that target the power consumption of the gate and make it independent of the original data by introducing noise and other leakage to equalize the power consumption.

While NTRU implementations with claimed resistance to power analysis attacks have been proposed, almost all these implementations are based on heuristic protection measure and do not provide well-founded assurance regarding the resistance of the implementation against this class of attacks. Furthermore, our proposal is the first countermeasure for hardware implementation of NRTU.

Several unprotected hardware implementations of NTRUEncrypt were previously presented (e.g., see [36], [3], [46], [73]). Techniques to strengthen hardware implementations of NTRUEncrypt against fault analysis attacks were presented in [37]. Lee *et al.* [45] presented power analysis attacks on software implementations of NTRUEncrypt where they presented two attacks, SPA and CPA, and proposed several countermeasures. They used a common convolution algorithm to calculate $a(x) = f(x) * e(x)$ where $f(x)$ is the secret and $e(x)$ is the ciphertext. The basic operations of the algorithm are shift and add. In their algorithm, the accumulation registers are all initialized to zero. To conduct an attack, they assume a difference in power consumption between addition with non-zero operands such as: $x + y$, $x, y \neq 0$ and addition

with zero operands such as: $x+0$ or $0+y$. Because the attacker can control the ciphertext polynomial in the chosen ciphertext attack scenarios, the attacker can set the ciphertext polynomial to have all nonzero coefficients. This will enforce all addition operations in the first iteration of the convolution algorithm to have addition with zeros which allows the attacker to recover secret key information by observing the power consumption of this process. To defend against SPA, Lee *et al.* proposed to initialize all registers to non-zero values; however, initializing to a fixed value remains vulnerable. Therefore, their only solution is to initialize the registers to random values at each run of the algorithm. In their CPA attack, the attacker relies on the correlation between processed data and power consumption. The proposed countermeasure to the CPA attacks involves randomizing the temporary data stored in accumulator registers (similar to the SPA countermeasure), blinding public data, and randomizing the secret data. Lee *et al.* also investigated several shortcomings of each approach.

3.4 Proposed Countermeasures

The IEEE P1363.1 standard presents a few typical parameter sets for NTRU. In this thesis, we focus on the set of parameters where q is in the form of 2^n because of the simplicity of the modular operations for such choices of q , where the effect of the modular operation can be achieved by simply truncating the results to n bits. Table 3.2 shows the (N, p, q) parameters presented in the original NTRU proposal [31, 32]. Throughout our implementation, we utilize the Mersenne primes method (see Algorithm 1) to calculate $a(x) \mod p$. In this method, $a(x)$ can be split into sections where each has a length of $log2(p+1)$ bits, and the addition of these sections is the $a(x) \mod p$.

Algorithm 1 Modular reduction using Mersenne primes algorithm

Input: an integer a, a Mersenne prime p
Output: $b = a \mod p$
1: $b = a$
2: do while $b > p$
3: split b into sections $s_i|s_{i-1}|...|s_i|s_0$ each of length $log_2(p+1)$ bits
4: $b = s_i + s_{i-1} + ... + s_1 + s_0$

Note that in order to increase the efficiency of the decryption implementation, we choose $f = 1+pF$ which is computed in one convolution for the entire decryption process as explained in the previous subsection. We also chose NTRUEncrypt parameters presented in [31] as shown in 3.2.

3.5 Proposed Decryption Architecture

Similar to other public key cryptosystems, since the secret key is not used in the encryption process, the encryption process does not need to be protected against power analysis attacks. Consequently, in what follows, we focus on protecting the decryption process.

Our proposed countermeasures for NTRU decryption are inspired from traditional masking schemes (e.g., see [62]). However, unlike other algorithms, masking for NTRUEncrypt cannot be carried out linearly due to the nature of NTRU and the complexity of interaction between two different polynomial rings, R_p and R_q. For example, for typical choices of p and q, we cannot perform the decryption by splitting the secret $f(x)$ into two uniform shares over R_p because the first convolution operation in the decryption process is performed in R_q. In other words, for $f'(x), f''(x) \in R_p$, $f'(x) + f''(x) \mod p = f(x)$ does not guarantee that $((f'(x) * e + f''(x)) * e) \mod p = (f(x) * e) \mod p$ because this convolution operation is performed in R_q and interchanging the order of the mod p and mod q operations leads to the wrong result. Furthermore, if we split the shares over Z_q, i.e., if we choose $f'(x), f''(x) \in R_q$ such that $f'(x) + f''(x) \mod q = f(x)$, then we cannot ensure the uniformity of the shares because the coefficients of $f(x)$ are chosen to be very small (e.g., usually we have $f_i \in \{+1, 0, -1\}$ for $p = 3$). In addition, when $f'(x), f''(x) \in R_q$ we lose the advantage of being able to efficiently implement the convolution associated with first operation in the decryption process since in this case, the coefficients of $f'(x), f''(x) \in Z_q$ instead of $\{1, -1, 0\}$.

On the other hand, it should be noted an attacker who measures the power consumption associated with the convolution operation cannot derive useful information about one input to

the convolution module if the second input is a uniformly generated random polynomial which is freshly generated for each convolution operation and is not under the control of the attacker. More precisely, if the attacker measures one trace of power consumption associated with the operation $f(x) * e(x)$ then the attacker cannot derive information about $f(x)$ if $e(x)$ is a random input chosen uniformly from R_q and changes after each convolution operation. Thus, in our implementation we split the ciphertext into two random shares $e', e'' \in R_q$ such that the addition of these shares $\mod q$ yields the original ciphertex, i.e., $e' + e' = e \in R_q$. The two ciphertext shares are then processed in parallel using the secret key to produce $a' = f(x) * e'(x)$ and $a'' = f(x) * e(x)''$.

In the second step of the decryption process, we avoid computing $a(x) = (a'(x) + a''(x)) \in R_q$ and then centerlifting the results to obtain $b(x) \in R_p$. The main reason for not doing this is that while the two shares $e'(x)$ and $e''(x)$ cannot be controlled by the attacker, $a'(x) + a''(x) = f(x) * (e'(x) + e''(x)) = f(x) * e(x)$ and $e(x)$ is still under the control of the attacker who can manipulate the choice of $e(x)$ to maximize the information obtained by measuring the power consumption associated with the step of evaluating $a(x) \mod q$. To avoid this problem, we continue the computation along both branches of the decryption process performing the computation in R_p. However, in general if $y, z \in Z_q$ then $((y + z) \mod q) \mod p \neq ((y \mod p) + (z \mod p)) \mod p$. The following Lemma is used in order to allow us to overcome this problem.

Lemma 1 *Let $y, z \in Z_q$. Then we have*

$$((y + z) \mod q) \mod p =$$

$$\begin{cases} ((y \textit{ mod } p) + (z \textit{ mod } p)) \textit{ mod } p, \ y + z < q \\ ((y \textit{ mod } p) + (z \textit{ mod } p) + (q \textit{ mod } p)) \textit{ mod } p, y + z \geq q. \end{cases}$$

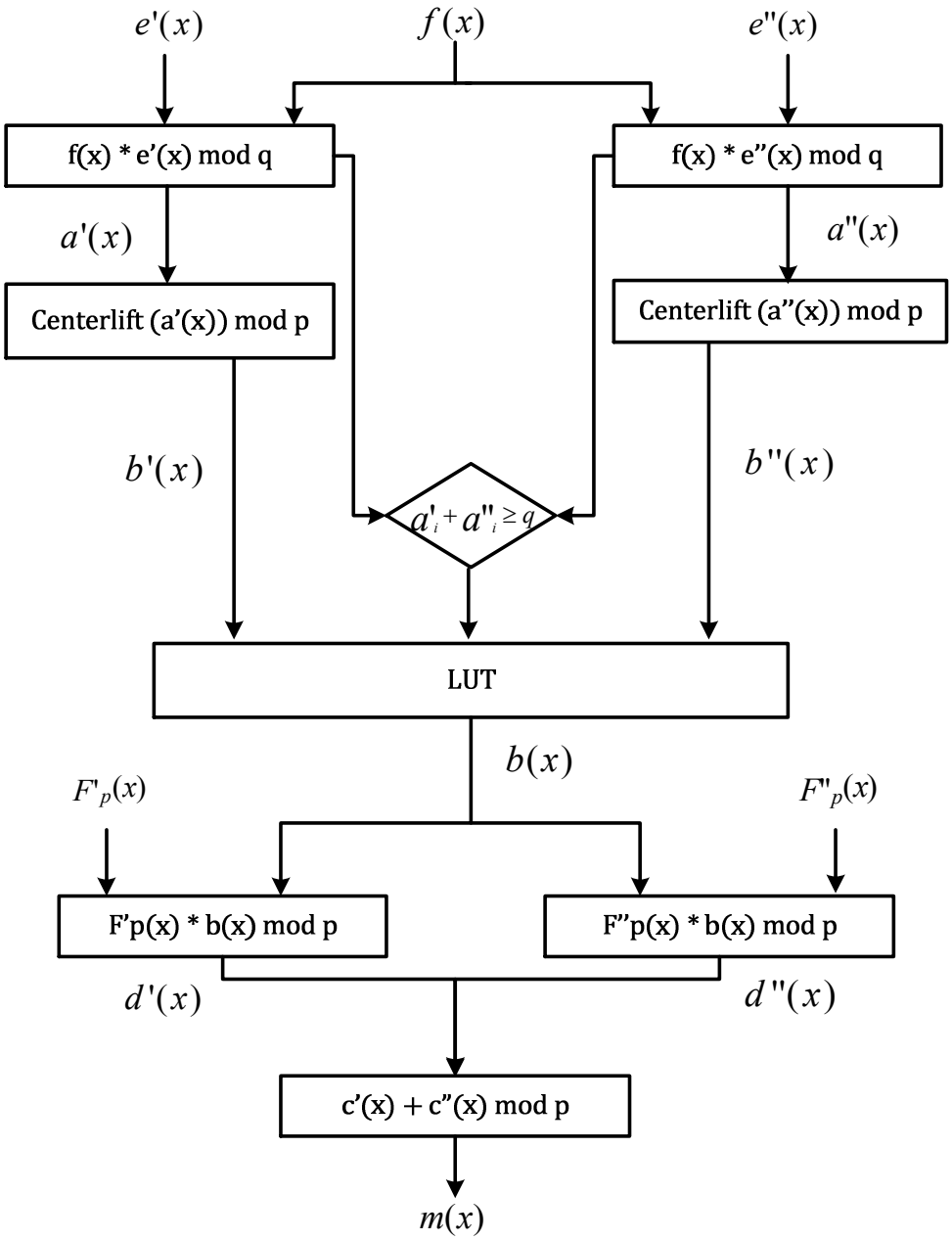


Figure 3.1: Top level view of the proposed masking scheme

Proof. The proof follows by noting that for $y, z \in Z_q$, we have $y < q$ and $z < q$. Hence we have

$$y + z \mod q = \begin{cases} y + z & y + z < q \\ y + z + q & y + z \geq q. \end{cases}$$

Fig.3.1 shows a top level view of our proposed architecture. As mentioned above, in the first stage of convolution, the ciphertext is arithmetically split into two random shares, e' and e'' in R_q where $e = e' + e'' \mod q$. The first two stages of the decryption process are computed with these shares as follows:

$$\begin{aligned} a'(x) &= e'(x) * f(x) \mod q \\ a''(x) &= e''(x) * f(x) \mod q \end{aligned} \tag{3.3}$$

Then we proceed to calculate $b(x) \in R_p$ without calculating $a(x) \in R_q$ as follows:

$$\begin{aligned} b'(x) &= a'(x) \mod p \\ b''(x) &= a''(x) \mod p \\ b(x) &= b(x)' + b''(x) + \gamma(x) \mod p \end{aligned} \tag{3.4}$$

where $\gamma(x) \in R_p$ is a correction polynomial evaluated using Lemma 1. More precisely, γ_i is set to $-q \mod p$ if $b'_i + b''_i \geq q$, and is set to zero otherwise, $i = 0, 1, \cdots, N-1$. In order to avoid revealing the coefficients of $\gamma(x)$ in the last step of equation (4.2) above by observing the difference in power consumption for the case of adding a zero versus adding $-q \mod p$, this step is performed using lookup tables (LUTs), where the inputs to each LUT are the corresponding coefficients of b', b'', and a binary input denoting whether $\gamma(x) = 0$ or $-q \mod p$. Thus, each LUT can be indexed by $(2 \times \lceil log_2(p) \rceil + 1)$ bits.

To evaluate $\gamma(x)$, we need to test whether $a'_i(x) + a''_i(x) > q$ for $i = 0, 1, \cdots, N-1$. In order to do so without explicitly evaluating $a'_i(x) + a''_i(x)$, we use the traditional carry-look-ahead method. Let a'_{i_j}, a''_{i_j} denote the j^{th} bit in the i^{th} coefficient of $a'(x)$ and $a''(x)$, respectively. Then, for $i = 0, 1, \cdots, N-1$, and $j = 0, 1, \cdots, (log_2(q) - 1)$, we evaluate:

$$\begin{aligned} c_{j+1} &= g_j \vee (p_j \cdot c_j), \\ g_j &= a'_{i_j} \cdot a''_{i_j} \\ p_j &= a'_{i_j} \vee a''_{i_j} \end{aligned} \tag{3.5}$$

where c_j is the carryout of the j^{th} bit. If $c_{log_2(q)} = 1$, then we conclude that $a'_i + a''_i > q$ and we need to add the correction factor.

In order to recover the message, $F_p(x)$ is also split into two different shares $F'_p(x), F''_p(x) \in R_p$ such that $F'_p(x) + F''_p(x) = F_p(x) \mod p$. Then the two shares are convoluted with $b(x)$ to generate $d'(x)$ and $d''(x)$, respectively as follows:

$$\begin{aligned} d' &= F'p(x) * b(x) \mod p \\ d'' &= F''p(x) * b(x) \mod p \end{aligned} \tag{3.6}$$

Finally, the message is recovered by adding $d'(x)$ and $d''(x)$ as follows:

$$m(x) = d'(x) + d''(x) \mod p \tag{3.7}$$

It should also be noted that our protection countermeasures are also applicable to other variants of NTRUEncrypt where the secret key $f(x)$ is chosen in the form $f(x) = 1 + pF(x)$. In this case, the last step of the decryption process is eliminated because $F_p = 1$ and hence $m(x) = b(x)$.

3.6 Results and Discussion

The proposed decryption architecture is implemented in hardware using Altera FPGA Cyclcone IV chip with Quartus II and ModelSim software. *Cyclone IV EP4CE115F29C7* is chosen as the target device in the provided implementation.

Figures 3.2 and 3.3 show the hardware architecture for the convolution operations in R_q and R_p, respectively. It takes N clock cycles to calculate both $a'(x)$ and $a''(x)$ and one clock cycle to calculate $b'(x)$ and $b''(x)$. Finally, it takes N clock cycles to calculate $d'(x)$, $d''(x)$ and then calculating $m(x)$. Thus, this architecture requires $2N + 1$ clock cycles to decrypt the ciphertext. Throughout our implementation, we utilize the Mersenne primes method [73] to

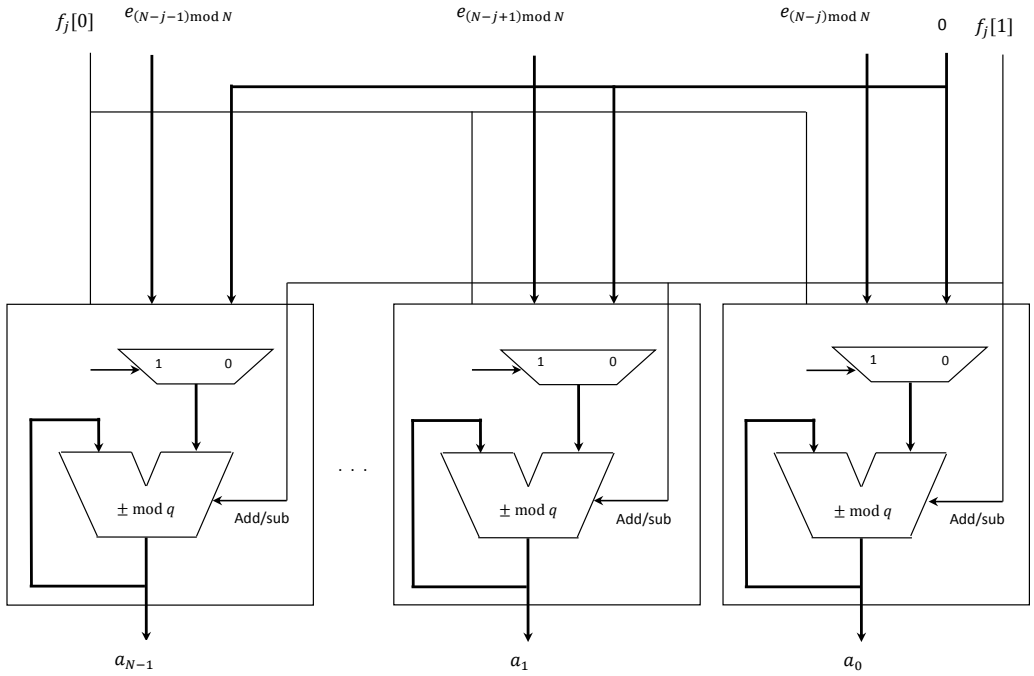


Figure 3.2: Circuit for evaluating $f * e \bmod q$

Table 3.3: Decryption without masking

(N, p, q)	**#LE**	**#RE**	**#CC**	**FMAX (MHZ)**	**Latency (μsec)**
$(1167, 3, 128)$	$6,142$	$3,459$	335	113.86	2.94
$(263, 3, 128)$	$9,556$	$5,359$	527	108.45	4.86
$(503, 3, 256)$	$20,512$	$11,165$	$1,007$	90.65	11.11

calculate $a(x) \mod p$.

The post-synthesis simulation results for decryption without any masking are shown in Table 3.3 where LE denotes the number of Logical Elements used in the device, RE denotes the number of registers, and CC denotes the number of clock cycles.

Table 3.4 shows the corresponding results for the protected implementation. As shown

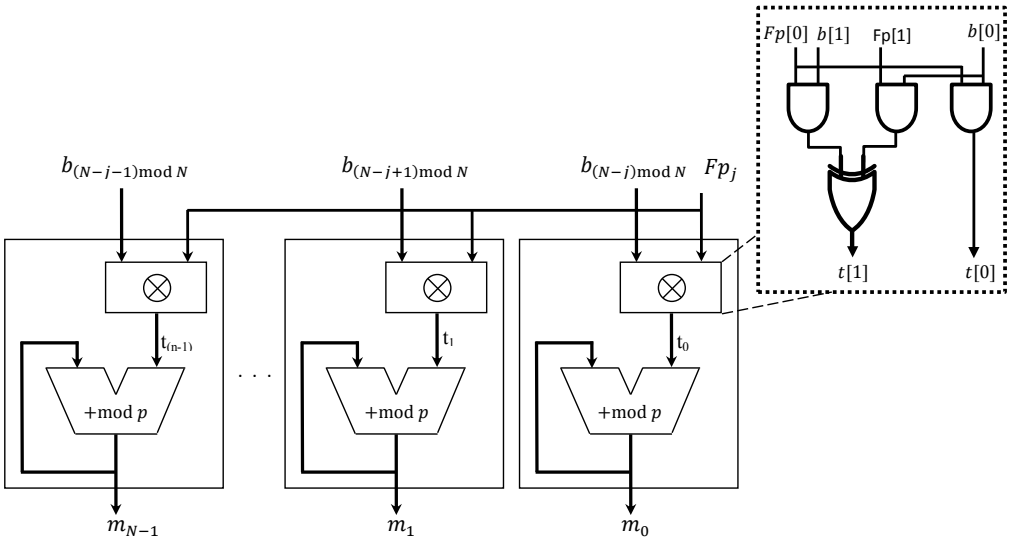


Figure 3.3: Circuit for evaluating the convolution multiplication $F_p * b \bmod p$

Table 3.4: Decryption with masking

(N, p, q)	**#LE**	**#RE**	**#CC**	**FMAX (MHZ)**	**Latency (μsec)**
$(1167, 3, 128)$	$10,071$ (64%)	$3,893$	335	112.37	2.98 (1.4%)
$(263, 3, 128)$	$15,452$ (61.7%)	$6,009$	527	105.59	4.99 (2.7%)
$(503, 3, 256)$	$32,282$ (57.4%)	$12,298$	$1,007$	84.79	11.88 (6.9%)

in this table, the latency is slightly affected by the protection measures, where it is increased by about 1.4%, 2.7% and 6.9% for the three considered set of parameters. On the other hand, the area overhead varies between 64%, to 57.4% for these parameters, which emphasizes the

nontrivial cost associated with protecting this class of cryptosystems against power analysis attacks.

Chapter 4

Rainbow Signature Scheme

4.1 Introduction

Rainbow is an MPKC public signature scheme from the Oil-Vinegar family which is based on solving multivariate equations over a finite field. The Oil-Vinegar family can be classified as balanced Oil-Vinegar, unbalanced Oil-Vinegar, and Rainbow. Rainbow is a construction of unbalanced multi-layer Oil-Vinegar for added security and efficiency [23].

4.1.1 Definitions

The finite field used is $GF(2^8)$, and it is denoted by k. The letters o and v represent the number of Oil and Vinegar variables respectively. Rainbow is a multiple layer Oil-Vinegar where the number of layers is denoted by u. Throughout the thesis the use of u represents the total number of layers used while l refers to a specific layer. The message (or hash of the message) is denoted by Y, and the signature is denoted by X. The signature size is n.

4.1.2 Oil-Vinegar

The key construction in the Oil-Vinegar family is a map F from k^{o+v} to k^o. This map is hidden between two affine linear transformations L_1 and L_2 as shown in the following set of

equations. L_1 and L_2 are on k^o and k^{o+v} fields, respectively. The map

$$\overline{F} = L_1 \circ F \circ L_2 \tag{4.1}$$

where

$$\begin{aligned} &F(x_1, ..., x_o, x'_1, ..., x'_v) \\ &= (F_1(x_1, ...x_o, x'_1, ..., x'_v), ..., F_o(x_1, ...x_o, x'_1, ..., x'_v)) \end{aligned} \tag{4.2}$$

and each F_l is in the form of

$$\begin{aligned} &F_l(x_1, ..., x_o, x'_1, ..., x'_v) \\ &= \sum a_{l,i,j} x_i x'_j + \sum b_{l,i,j} x'_i x'_j + \sum c_{l,i} x_i + \sum d_{l,j} x'_j + e_l \end{aligned} \tag{4.3}$$

where $x_i, i = 1, ..., o$ are the Oil variables and $x'_j, j = 1, ..., v$ are the Vinegar variables in the finite field k as shown in [23].

In Equation 4.3, there are terms where the Oil and Vinegar variables do not mix; this inspired the Oil-Vinegar name. The balanced Oil-Vinegar is when $v = o$. However, it was broken by Kipnis and Shamir [39]. The unbalanced Oil-Vinegar, on the other hand, is when $v > o$; however, to ensure security the size of v and o has to be carefully selected [23]. Rainbow is a construction of unbalanced Oil-Vinegar in multiple layers for added security and efficiency.

The number of Oil variables in the i^{th} layer, o_i, and the number of Vinegar variables, v_i, are related as follows: $v_{i+1} = o_i + v_i$. This relation means that Vinegar variables in the next $i^{th}+1$ layer is the concatenation of the previous layer of Oil and Vinegar variables. The l^{th} layer consists of o_l polynomials where $\{x_i \mid\in O_l\}$ denotes the set of Oil variables and $\{x'_j \mid\in S_l\}$ denotes to the set of Vinegar polynomials. The private secrets are $a_{l,i,j}, b_{l,i,j}, c_{l,i}, d_{l,j}$, and e_l, which means that each layer and each polynomial has its own sets of secrets.

4.2 Overview of Rainbow Scheme

In this section, we provide an overview of the Rainbow signature scheme [23].

Public Key

The public key of Rainbow consists of the $n - v_1$ polynomial components of $\overline{F}$ and the field structure of k.

Private Key

The private key consists of the maps $L1$, L_2, and F.

Signature Generation

To sign a document, which is an element $Y = (y_1, y_2, ..., y_{n-v_1}) \in k^{n-v_1}$, we find the solution of the following equation

$$L_1 \circ F \circ L_2(x_1, ..., x_n) = \overline{F}(x_1, ..., x_n) = Y \tag{4.4}$$

Then we apply the inverse of L_1 to get the following equation

$$F \circ L_2(x_1, ..., x_n) = L_1^{-1}Y = \overline{Y} \tag{4.5}$$

Next we invert F by solving the equation

$$F(x_1, ..., x_n) = \overline{Y} = (\overline{y}_1, ..., \overline{y}_{n-v_1}) \tag{4.6}$$

The values $x_1, ..., x_{v_1}$ are randomly chosen and plugged into the first layer of o_l equations given by

$$\overline{F}_1 = (\overline{y}_1, ..., \overline{y}_{o_1}) \tag{4.7}$$

The solution is plugged into the second layer of polynomials which will produce o_2 equations. The procedure is repeated until a solution is found. If at any layer, a solution is not found, we start from the beginning again by choosing another set of values for $x_1, ..., x_{v_1}$. We finally apply

the inverse of L_2, to obtain a signature of Y denoted by $X = (x_1, ..., x_n)$

Signature Verification

The verification is done by checking if the following equation holds

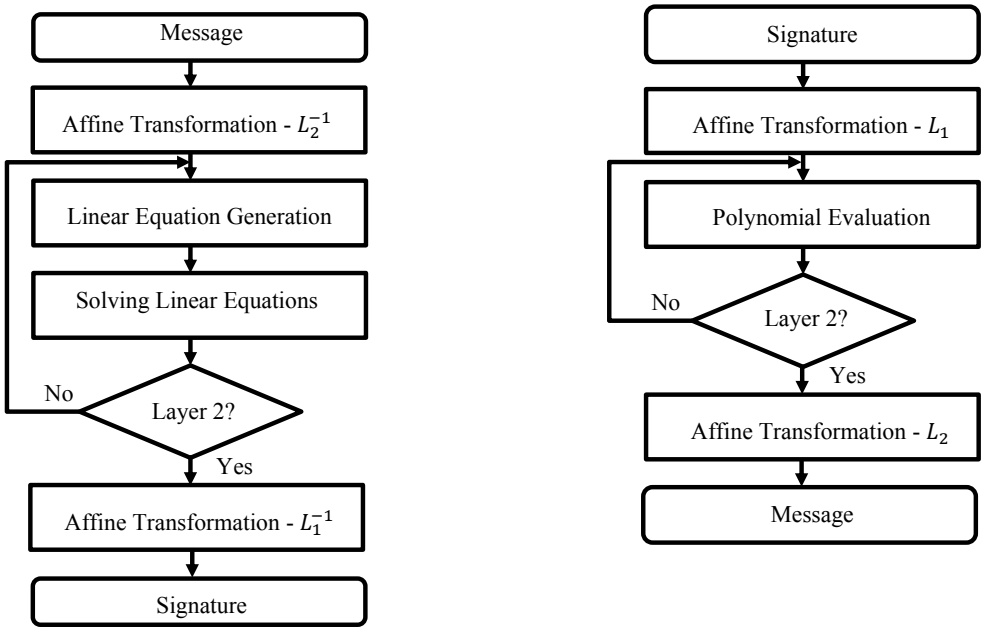
$$\overline{F}(X) = Y \tag{4.8}$$

Parameters for Rainbow

The parameters used in this work are similar to [23, 70]. We use a two layer Rainbow, $u = 2$, where the security level is above 2^{80}. The message size is $n = 24$ bytes while the signature size is 42 bytes. In the first layer, we use 17 Vinegar variables and 12 Oil variables while in the second layer we use 30 Vinegar variables and 12 Oil variables. One extra random Vinegar variable is generated to complete the set of 30 Vinegar variables in the second layer.

4.3 Related Work

The cryptanalysis for Rainbow has been considered since its introduction. Rainbow, unlike others, survived several known attacks such as the Grobner basis attacks [27], the rank attacks [39], and the differential analysis attacks [60]. In [29], Hashimoto *et al.* described a fault attack scheme that recovers parts of the secrets (private keys) in Rainbow by inducing a fault in the central map, F. The generated signature is a faulty signature named F'. The sparseness of $F - F'$ reveals some information about the private key. Another fault is injected into another element of F to get another faulty signature, F''. Again, the sparseness of $F - F''$ reveals other information about the private keys. The process is repeated several times to recover parts of the private key. In this thesis, we propose two approaches to protect against this type of fault analysis attack.



(a) Signature Flowchart

(b) Verification Flowchart

Figure 4.1: Top Level Rainbow Scheme

4.4 Hardware Implementation of Rainbow

The top level overview of Rainbow's signature and verification is shown Figure 4.1. The flowcharts show three major blocks: affine linear transformation, polynomial evaluation, and solving linear equations using the Gaussian elimination method. The hardware efficiency of the three blocks depends highly on the $GF(2^8)$ multiplication and partial multiplicative inversion.

Similar to [70], in this work, we implement the $GF(2^8)$ operations using the irreducible polynomial $x^8+x^6+x^3+x^2+1$ in order to improve the efficiency of the $GF(2^8)$ multiplication. The following subsections explain both schemes along with the improved implementation.

4.4.1 Improved Architecture

As mentioned above, both the signature and verification schemes depend highly on $GF(2^8)$ multiplication and inversion operations. We use a Look-Up Table (LUT) for the inverse and a generic implementation of Shift/Add for the multiplication process. We use highly parallel architecture for all units in the design: linear transformations L_1^{-1} and L_2^{-1}, polynomial evaluation, and Gaussian elimination process. In designing the latter, we base our implementation on methods used in [10, 70]. However, we noticed that the critical path is in the linear equations generation unit; the unit that generates linear equations from substituting the set of Vinegar variables in Equation 4.3. For this unit, we use highly parallel architecture where we compute the multiplications of independent vectors simultaneously. We only use this parallelism in the critical path which is the computation of components a and b and not c and d. Contemplating on the results, one sees that our Gaussian elimination process does not outperform Tang *et al.* implementation, but because of our improved linear equation generation unit design, we achieve a higher overall performance.

4.4.2 Signature Process

The block diagram of the signature process is shown in Figure 4.2. The controller unit controls the flow of data according to the flowchart. There are two layers in our Rainbow scheme, and each layer generates a set of 12 linear equations. Each one of these equations is generated with its own set of private keys. The total number of keys is shown in Table 4.1. The total number of secrets to protect in the central map, F, is $= 21,912$. Once the linear equations are generated, we forward them into the Gaussian elimination unit. The controller first solves layer 1 equations. If a solution exists, the solution (a set of Oil variables) is concatenated with the set of Vinegar variables, along with one randomly generated variable, and forwarded to the second layer. Otherwise, i.e. if no solution is found, another set of random Vinegar variables is chosen and the process is repeated. The 30 new Vinegar variables are substituted into Equation 4.3 to get another set of 12 linear equations. Layer 2 equations are solved and

Table 4.1: Private Keys (Secrets) of Rainbow

Type	Matrix Size	Total Number
L_1 Transformation	42×42	$1,764$
Layer 1 Polynomials	$o = 12, v = 17$	$6,276$
Layer 2 Polynomials	$o = 12, v = 30$	$15,636$
L_2 Transformation	24×24	576

a solution is generated. The solution is forwarded to the linear transformation stage. In this linear transformation, we use highly parallel architecture where we compute the multiplications of independent vectors simultaneously, similar to the linear equation generation unit.

4.4.3 Verification Process

The verification process, as shown in Figure 4.3, is a direct implementation of Equation 4.1. Linear transformation L_1 is followed by the two layers evaluations of Equation 4.3 and another linear transformation L_2. The same techniques of improving the critical path's performance used in the linear equation generation unit are applied to the polynomial evaluation unit.

4.4.4 Paralleled Gauss-Jordan Elimination

The systematic repetitive process of Gaussian elimination makes it easy to implement in hardware. Our implementation is based on highly parallel architecture where several tasks are executed in the same cycle. The process is composed of two steps: obtaining the upper triangle matrix and back substitution. The algorithm used is similar to Tang *et al.* [70] which is based on Bogdanov *et al.* [10]. The size of matrix is only 12×12 because $o = 12$. The solution is obtained by performing: pivoting, inversion, normalization, and elimination for each row, i, of the matrix. The algorithm presented below is designed to complete in $2 \times n$ cycles as shown in the next section.

The first step in solving a 12×12 matrix is pivoting which is the process of fetching the

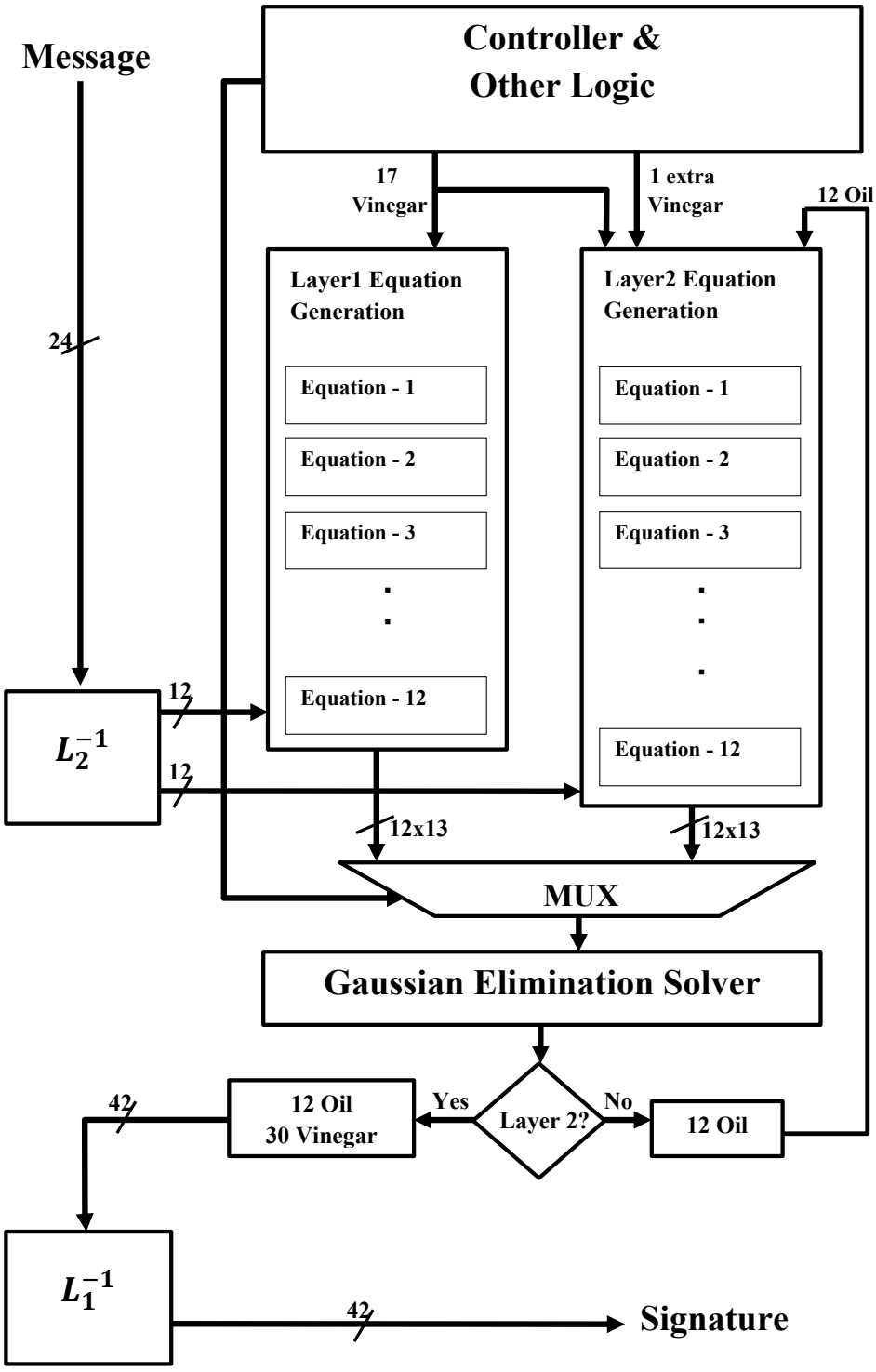


Figure 4.2: Signature Block Diagram

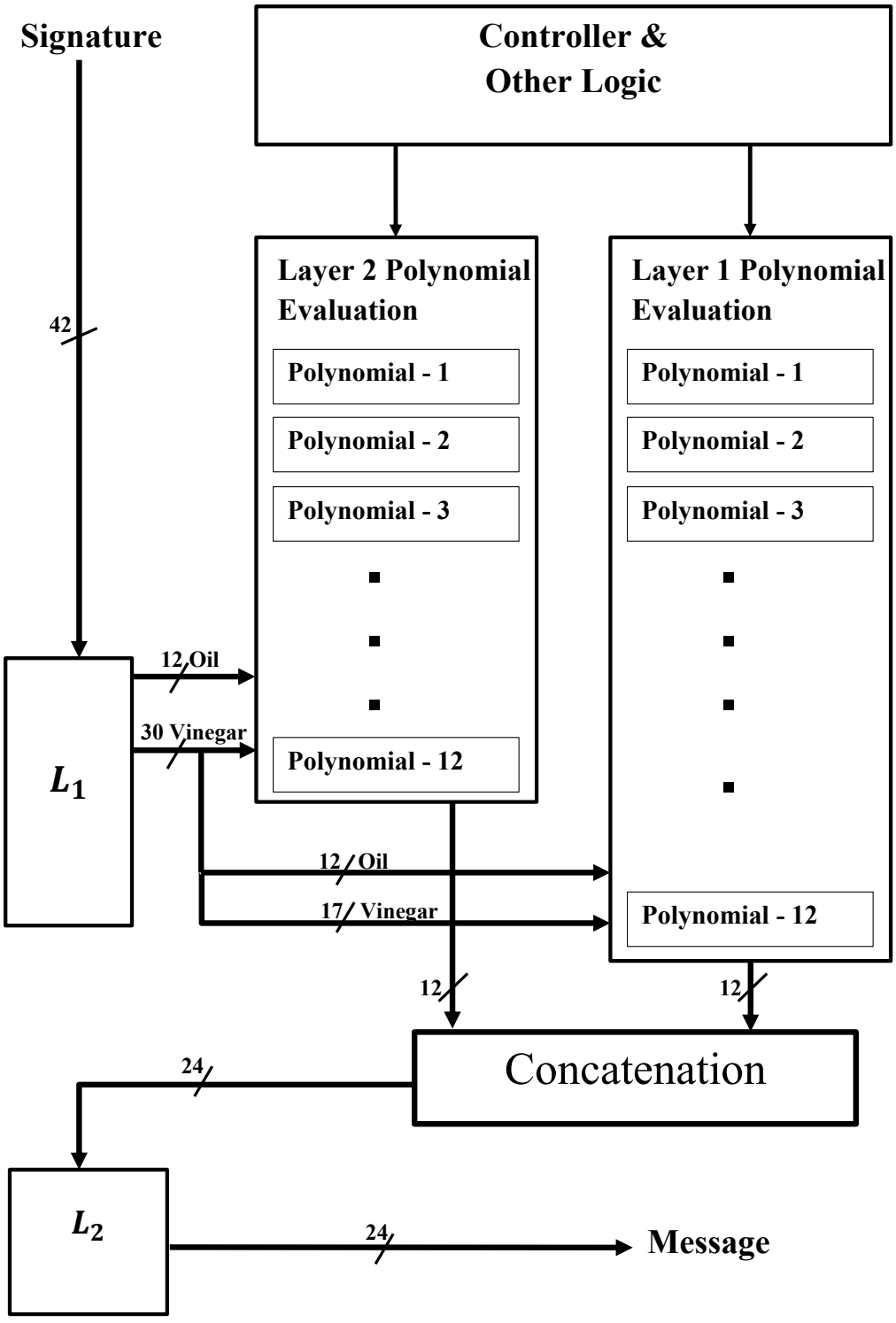


Figure 4.3: Verification Block Diagram

Algorithm 2 Gaussian Elimination Algorithm

```
for i = 1 to 12 do
    Pivoting(i)
    Partial Inversion(i)
    Normalization(i)
    Elimination(i)
end for
return Solution
```

leading non-zero element in the pivoting column of the matrix. Knowing that a_{ij} is an element in the i^{th} row and j^{th} column, the first column in the matrix is the pivoting column for $i = 0$. Once, the pivot element is recognized, we swap the matrix rows to ensure the pivot element is in the leading i^{th} row of the i^{th} iteration.

The inversion process is important for the followed elimination and normalization steps where the inverse of the pivot element a_{ij} is computed to obtain $1/a_{ij}$. We use a LUT for the finite field inversion of the size of $256\ bytes$ which is $8 \times 256 = 2,048\ bits$ since each of the $256\ GF$ elements occupies 8 bits.

Normalization is the process of dividing the leading row, i.e., the i^{th} row, of the i^{th} iteration by the pivot element. This is done by multiplying the pivot row by the inverse value of a_{ij}. To obtain the triangle upper matrix, normalization and elimination are repeated for each row after pivoting.

Elimination is performed for each row of the matrix after normalization. Each subsequent row following the normalized row is subtracted from it to obtain an upper triangle matrix.

The described four processes are all repeated for each row of the matrix, i.e., for 12 iterations. Once the upper triangle matrix is obtained, the back substitution process is computed in one cycle using a highly parallel architecture of vectors multiplication and addition.

4.4.5 Affine Linear Transformation

The linear transformations L_1^{-1} and L_1 for the signature and verification respectively are transformations in the same field k^{24} to k^{24}, while the transfer L_2^{-1} and L_2 are from k^{42} to k^{42}. Both transformations are computed using highly parallel architecture of vectors multiplications

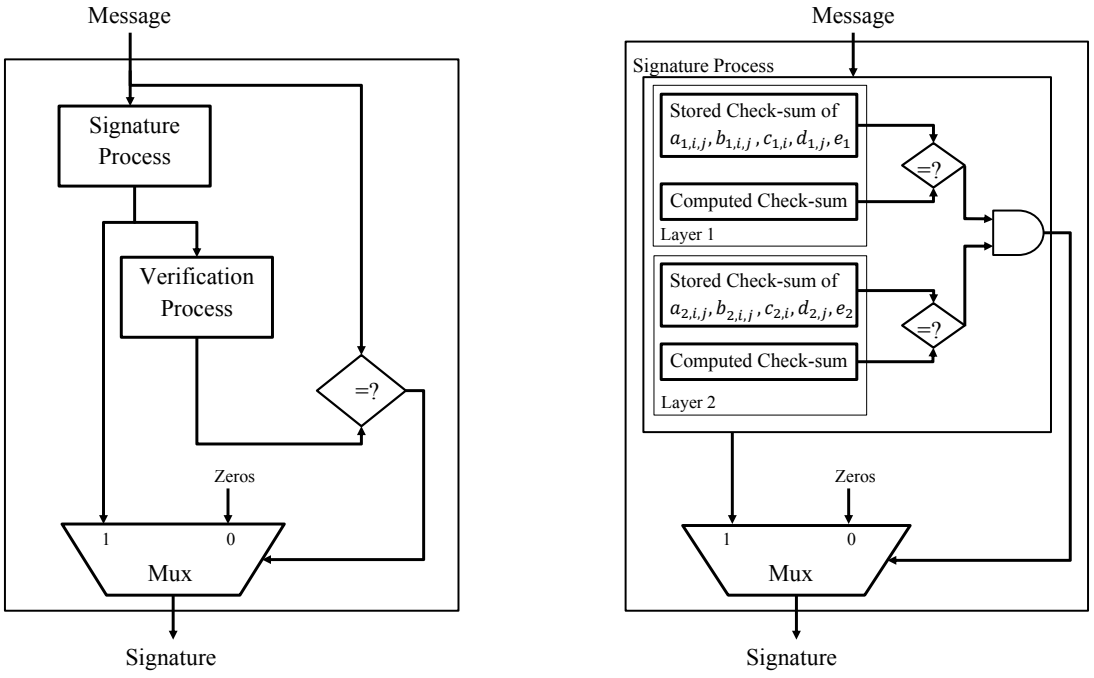
and additions.

4.5 Proposed Fault Analysis Countermeasures

The fault attack presented in [29], as described Section I above, succeeds if the attacker is able to inject a fault in the central map, F. This fault causes a faulty signature F' to occur. The sparseness of the original signature and the faulty signature $F - F'$ reveals information about the private key. By repeating this process, the attacker can gradually reveal more information about the secret key. To protect against this attack, we first considered a spacial redundancy solution where two parallel signatures are implemented side by side on the same chip. If both signatures match, then we output the signature; otherwise, the chip output is disabled. Because this approach is very expensive in terms of area, we abandoned it and moved to other solutions. Similarly, we did not implement the straightforward temporal redundancy scheme, where the signature operation is repeated twice and the output of the two runs are compared, because of the large time overhead.

The most intuitive approach to detect faults in the Rainbow signature algorithm is to have a verification unit checking the results of the signature process on the same chip. We call this approach, the signature verification scheme, see Figure 4.4a. In this approach, we feed the output of the signature process into the verification unit and compare the results with the original message. If the original message is retrieved, i.e. Equation 4.8 holds, then we can safely assume that no tampering of the private secrets occurred under the assumption that the attacker cannot inject a fault $a_{l,i,j}$ in the same layer or the same linear transformation in both the signature and verification units. Indeed, this full verification countermeasure protects against all sorts of faults that can be injected in the signature process including the ones that can occur in the central map F as presented in [29].

In contrast to our signature verification scheme where we verify the entire signature process, if one aims only to protect the signature algorithm against the fault attack in Hashimoto



(a) Signature Verification

(b) Matrix Check-sum Verification

Figure 4.4: Two Proposed Fault-resistant Schemes

et al. [29] which injects one fault at a time in the central map keys, then protecting central map keys F only is sufficient to prevent this attack. We propose protecting the central map keys, F by computing the check-sum of each row and each column of the matrices corresponding to the secret coefficients in Equation 4.3. We call this approach, the check-sum verification scheme, see Figure 4.4b. If the computed check-sum results match the stored ones, we conclude that no fault attacks occurred under the assumption that the attacker cannot inject a fault into the private keys and its corresponding stored check-sum matrices.

4.6 Results

The signature, verification, and check-sum are all implemented on Xilinx Vertix 7 family, $XC7VX980T$. The signature and verification are compared to Tang *et al.* [70] because of

Table 4.2: Signature Comparison with Tang *et al.* [70]

Steps	**Components**	**Clock Cycle**	
		Tang *et al.* [70]	Ours
1	L_2^{-1} Transformation	5	16
2	The first 12 polynomials evaluations	45	38
3	The first round of solving system linear equations	12	25
4	The second 12 polynomials evaluations	111	64
5	The second round of solving system linear equations	12	25
6	L_1^{-1} Transformation	13	23
	Total	198	191

Table 4.3: Verification process Timings

Steps	**Components**	**Clock Cycles**
1	L_1 Transformation	24
2	The first 12 polynomials evaluations	64
3	The second 12 polynomials evaluations	38
4	L_2 Transformation	15
	Total	141

the same Rainbow parameters and clock frequency. We both use a $20ns$ clock period; however, their implementation is based on a Xilinx Startix II chip while we use a Vertix 7 family chip. The latter is a bigger chip and can contain the signature unit as well as the proposed countermeasures. Indeed, our implementation is a slight improvement of their implementation as shown in Table 4.2 especially in the linear equation generation stages. Even though our Gaussian elimination unit executes in $2 \times n$ cycles and Tang *at al.* [70] executes in only n cycles, our overall performance is better than theirs. This is due to the improved parallel architecture in the critical paths. For the verification process, our results are summarized in Table 4.3.

The results of fault attack-resistant schemes are listed in Table 4.4. The signature verification approach increases the required area by about 33% while the check-sum verification increases the area by 9%. The number of cycles presented in Table 4.2 and Table 4.4 are based on finding Gaussian elimination solutions from the very first attempt, which will happen with

Table 4.4: Number of slices and clock cycles for the fault-resistant implementations

Scheme	**#Slices (% of chip)**	**Clock Cycle**
Signature alone $L_2^{-1} \circ F \circ L_1^{-1}$	$78,837(51\%)$	200
Verification alone $L_1 \circ F \circ L_2$	$39,486(25\%)$	141
Signature Verification Scheme	$105,202(68\%)$	344
Check-sum Verification Scheme	$85,630(55\%)$	200

a very high probability given the chosen set of parameters [23]. Furthermore, the number of cycles for the signature alone is listed to be 200 while it is listed 191 in Table 4.3. The extra cycles are mainly due to the time taking to generate the random variables, which we perform using a set of LFSRs. In a more secure hardware implementation, this step should be performed using on-chip true random number generators. The signature verification fault analysis-resistant scheme adds area and time to the design while the check-sum verification scheme adds only area to the design.

Chapter 5

Conclusions

5.1 Summary and Conclusions

This section briefly summarizes the accomplished work and the major contributions of our thesis. In chapters 1, 2, the essential background, mathematical assumptions, and motivation for this work were presented.

In chapter 3, we presented a masking scheme to protect NTRUEncrypt from first order differential power analysis attacks. It should also be noted that our protection countermeasures are applicable to other variants of NTRUEncrypt where the secret key $f(x)$ is chosen in the form $f(x) = 1 + pF(x)$. In this case, the last step of the decryption process is eliminated because $F_p = 1$. However, there will be a slight increase in the cost of implementing the convolution circuit that computes $e(x) * f(x)$ because in this case the coefficients of $f(x) \in \{1, -1, 0, 2\}$

In an unprotected NTRUEncrypt decryption, the recovered plaintext is computed by first performing convolution polynomial multiplication on the secret key modulo q and then performing convolution polynomial multiplication on secret inverse modulo p. We mask the polynomial operations by splitting the ciphertext polynomial and F_p polynomial into two random shares for each; and continuing in the masking each step of the decryption process until recovery of plaintext. Thus, the masking scheme keeps all intermediates in the masked domain.

In chapter 4, we presented an improved hardware implementation of the Rainbow scheme along with two fault analysis countermeasures. The first approach protects the private keys of both linear transformations and central map, F, of the Rainbow scheme under the assumption that the attacker cannot inject the same fault in both the signature and verification units. The second approach, on the other hand, protects only the central map keys under the assumption that the attacker cannot inject faults into both private key matrices and its corresponding check-sum matrices. We compared both implementations and showed that signature verification fault analysis-resistant approach increases time by a factor of 72% while the check-sum verification approach does not add any time overhead. In addition, the area penalty for the first is 33% while the area penalty of the latter is only 9%.

5.2 Future Work

In what follows we list some of the topics that can be a future extension to the projects shown in this thesis.

- We can investigate more options to achieve a secure hardware implementation for the NTRUEncrypt. For example, we can consider the circuit level solutions for preventing DPA attacks such as the threshold implementation.

- The proposed countermeasures for Rainbow can be suitable for other post-quantum algorithms. Thus, we can apply these ideas to other cryptosystems.

- Rainbow signature scheme is very well studied, and proved to be secure. The presented hardware implementation in this thesis is not optimal, because the focus of this thesis is to propose secure hardware schemes to prevent fault analysis attack presented in Hashimoto *et al.* and not to present an efficient hardware implementation. A more efficient hardware implementation can be researched and studied.

- Side Channel Attacks target the hardware implementation of an algorithm for inadvertent

leakages during running time. A recent research in the field of leakage-resilient cryptography focuses on the design of cryptographic primitives resistant to arbitrary SCAs. In SCAs, the attacker collects large amount of information on the private key. However, these large amount of information are bounded by some parameter. A future work will be to study the leakage-resilient cryptography and find methodologies to achieve bounded leakage.

- For the NTRUEncrypt we can explore other approaches, such as the use of threshold implementations, to enhance the area overhead and reduce time penalty.

Bibliography

[1] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov. Cryptographic Processors-A Survey. *Proceeding of the IEEE*, 94(2):357–369, Feburary 2006.

[2] M. Arora. How secure is AES against brute force attacks? `https://www.eetimes.com/document.asp?doc_id=1279619`, Novemeber, 2013. EE Times.

[3] D. V. Bailey, D. Coffin, A. Elbirt, J. H. Silverman, and A. D. Woodbury. NTRU in Constrained Devices. In *International Workshop on Cryptographic Hardware and Embedded Systems CHES 2001: Cryptographic Hardware and Embedded Systems CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 262–272, Springer, Berlin, Heidelberg, 2001. Springer.

[4] S. Balasubramanian, A. Bogdanov, A. Rupp, J. Ding, and H. W. Carter. Fast Multivariate Signature Generation in Hardware: The Case of Rainbow. In 16^{th} *International Symposium on Field-Programmable Custom Computing Machines. FCCM '08*, pages 281–282. IEEE, April 14-15 2008.

[5] E. R. Berlekamp. Goppa Codes. *IEEE Transactions on information theory*, IT-19(5): 773–776, September 1973.

[6] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer, USpringer-Verlag Berlin Heidelberg, 2009.

[7] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Pa-

pachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical Stateless Hash-Based Signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2015: Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397, Springer, Berlin, Heidelberg, 2015. Springer.

[8] D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *International Workshop on Post-Quantum Cryptography PQCrypto 2008: Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46, Springer, Berlin, Heidelberg, 2008. Springer.

[9] J. Blömer, J. Guajardo, and V. Krummel. Provably Secure Masking of AES. In *International Workshop on Selected Areas in Cryptography SAC 2004: Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83, Springer, Berlin, Heidelberg, 2004. Springer.

[10] A. Bogdanov, M. Mertens, C. Paar, J. Pelzl, and A. Rupp. A Parallel Hardware Architecture for fast Gaussian Elimination over GF(2). In 14^{th} *Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 237–248. IEEE, April 24-26 2006.

[11] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[12] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In *IMA International Conference on Cryptography and Coding IMACC 2013: Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64, Springer, Berlin, Heidelberg, 2013. Springer.

[13] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *International Workshop on Cryptographic Hardware and Embedded Systems CHES 2004: Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29, Springer, Berlin, Heidelberg, 2004. Springer.

[14] J. Buchmann, E. Dahmen, and A. Húlsing. XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In *Annual International Cryptology Conference CRYPTO 2001: Advances in Cryptology (CRYPTO 2001)*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129, Springer, Berlin, Heidelberg, 2001. Springer.

[15] D. Butin. Hash-Based Signatures: State of Play. *IEEE Security & Privacy*, 15(4):37–43, August 2017.

[16] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Annual International Cryptology Conference CRYPTO 1999: Advances in Cryptology (CRYPTO 99)*, volume 1666, pages 398–412, Springer, Berlin, Heidelberg, 1999. Springer.

[17] L. Chen. Cryptography Standards in Quantum Time: New Wine in an Old Wineskin? *IEEE Security & Privacy*, 15(4):51–57, August 2017.

[18] M. Ciet and M. Joye. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. *Designs, Codes and Cryptography*, 36(1), July 2005.

[19] D. Coppersmith, J. Stern, and S. Vaudenay. Attacks on the Birational Permutation Signature Schemes. In *Annual International Cryptology Conference CRYPTO 1993: Advances in Cryptology (CRYPTO'93)*, volume 773 of *Lecture Notes in Computer Science*, pages 435–443, Springer, Berlin, Heidelberg, 1993. Springer.

[20] CrytoMathCREST. Mathematical Modeling for Next-Generation Cryptography. `https://cryptomath-crest.jp/english/`, 2014. Online.

[21] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):664–654, November 1976.

[22] J. Ding and A. Petzoldt. Current State of Multivariate Cryptography. *IEEE Security & Privacy*, 15(4):28–36, August 2017.

[23] J. Ding and D. Schmidt. Rainbow, a New Multivariable Polynomial Signature Scheme. In *International Conference on Applied Cryptography and Network Security ACNS 2005: Applied Cryptography and Network Security pp 164-175*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2005.

[24] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice Signatures and Bimodal Gaussians. In *Annual International Cryptology Conference CRYPTO 2013: Advances in Cryptology (CRYPTO 2013)*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56, Springer, Berlin, Heidelberg, 2013. Springer.

[25] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.

[26] J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. In *IACR Cryptology ePrint Archive*, https://eprint.iacr.org/2012/144, 2012. IACR.

[27] J. C. Faugére. A New Efficient Algorithm for Computing Gröbner Bases F_4. *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[28] J. C. Faugére, F. coise Levy-dit Vehel, and L. Perret. Cryptanalysis of MinRank. In *Annual International Cryptology Conference CRYPTO 2008: Advances in Cryptology (CRYPTO2008)*, volume 5157 of *Lecture Notes in Computer Science*, pages 280–296, Springer, Berlin, Heidelberg, 2008. Springer.

[29] Y. Hashimoto, T. Takagi, and K. Sakurai. General Fault Attacks on Multivariate Public Key Cryptosystems. In *IEICE Transactions on Fundamentals of Electronics, Communications*

and Computer Sciences, volume E96-A of *Lecture Notes in Computer Science*, pages 196–205. Springer, January 2013.

[30] J. Hoffstein, J. Pipher, and J. Silverman. *An Introduction to Mathematical Cryptography*. Undergraduate Texts in Mathematics. Springer, Springer-Verlag New York, 2008.

[31] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A Ring-based Public Key Cryptosystem. In *International Algorithmic Number Theory Symposium ANTS 1998*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[32] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A public key cryptosystem. Avilable at http://grouper.ieee.org/groups/1363/lattPK/ submissions/ntru.pdf, 1999.

[33] IBM Press. IBM Builds Its Most Powerful Universal Quantum Computing Processors. `https://www-03.ibm.com/press/us/en/pressrelease/52403.wss`, May, 2017. IBM, Yorktown Heights, N.Y.

[34] D. Jao and L. D. Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *International Workshop on Post-Quantum Cryptography PQCrypto 2011: Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34, Springer, Berlin, Heidelberg, 2011. Springer.

[35] M. Joye, A. K. Lenstra, and J.-J. Quisquater. Chinese Remaindering Based Cryptosystems in the Presence of Faults. *Journal of Cryptology*, 12(4), September 1999.

[36] A. Kamal and A. M. Youssef. An FPGA Implementation of The NTRUEncrypt Cryptosystem. In *Proceedings of the International Conference on Microelectronics (ICM'09)*, pages 209–212, Marrakech, Morocco, 2009. IEEE.

[37] A. Kamal and A. M. Youssef. Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks. *Cryptographic Engineering*, 3(4):227–240, November 2013.

[38] A. Kipnis, J. Patarin, and L. Goubin. Unbalanced Oil and Vinegar Signature Schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 1999: Advances in Cryptology EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222, Springer, Berlin, Heidelberg, 1999. Springer.

[39] A. Kipnis and A. Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Annual International Cryptology Conference CRYPTO 1999: Advances in Cryptology (CRYPTO99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30, Springer, Berlin, Heidelberg, 1999. Springer.

[40] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[41] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Annual International Cryptology Conference CRYPTO 1999: Advances in Cryptology (CRYPTO 99)*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Springer, Berlin, Heidelberg, 1999. Springer.

[42] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Annual International Cryptology Conference CRYPTO 1996: Advances in Cryptology (CRYPTO'96)*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Springer, Berlin, Heidelberg, August 1996. Springer.

[43] L. Lamport. Constructing digital signatures from a one-way function. *Technical Report SRI-CSL-98*, 1, October 1979.

[44] K. Lauter. Postquantum Opportunities: Lattices, Homomorphic Encryption, and Supersingular Isogeny Graphs. *IEEE Security & Privacy*, 15(4):22–27, August 2017.

[45] M. Lee, J. E. Song, D. Choi, and D.-G. Han. Countermeasures against Power Analysis Attacks for the NTRU Public Key Cryptosystem. *IEICE Transactions on Fundamentals of*

Electronics, Communications and Computer Sciences, Volume E93.A, Issue 1, pp. 153-163 (2010)., E93.A(1):153–163, 2010.

[46] B. Liu and H. Wu. Efficient Architecture and Implementation for NTRUEncrypt System. In *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, Fort Collins, CO, USA, 2-5 August 2015. IEEE.

[47] V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2010: Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, Springer, Berlin, Heidelberg, 2010. Springer.

[48] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien. Experimental Realization of Shor's Quantum Factoring Algorithm using qubit Recycling. *International Workshop on Post-Quantum Cryptography PQCrypto 2008: Post-Quantum Cryptography*, 6:773–776, February 2012.

[49] T. Matsumoto and H. Imai. Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques EUROCRYPT 1988: Advances in Cryptology EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453, Springer, Berlin, Heidelberg, 1988. Springer.

[50] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *DSN Progress Report*, 42(44), Januray and Feburary 1978.

[51] R. C. Merkle. *Secrecy, Authentication and Public Key Systems / A Certified Digital Signature*. PhD thesis, Dept. of Electrical Engineering, Stanford University, 1979.

[52] R. C. Merkle. A Certified Digital Signature. In *Annual International Cryptology Con-*

ference CRYPTO 1989: Advances in Cryptology (CRYPTO 89), volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Springer, New York, NY, 1989. Springer.

[53] M. Nakkar, M. Mahmoud, and A. Youssef. Fault Analysis-resistant Implementation of Rainbow Signature Scheme. In *Proceedings of the* 29^{th} *International Conference on Microelectronics*, Beirut, Lebanon, 2017. IEEE.

[54] P. Q. Nguyen and O. Regev. Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT 2006: Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288, Springer, Berlin, Heidelberg, 2006. Springer.

[55] P. Q. Nguyen and O. Regev. Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. *Journal of Cryptography*, 22(2):139–160, April 2009.

[56] H. Niederreiter. Knapsack Type Cryptosystems and Algebraic Coding Theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[57] D. A. Osvik, A. Shamir, and E. Tromer. Cache Attacks and Countermeasures: The Case of AES. In *Cryptographers Track at the RSA Conference CT-RSA 2006: Topics in Cryptology CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20, Springer, Berlin, Heidelberg, 2006. Springer.

[58] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *International Workshop on Fast Software Encryption FSE 2005: Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423, Springer, Berlin, Heidelberg, 2005. Springer.

[59] J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt88. In *Annual International Cryptology Conference CRYPTO 1999: Advances in Cryptol-*

ogy (CRYPTO 95), volume 963 of *Lecture Notes in Computer Science*, pages 248–261, Springer, Berlin, Heidelberg, 1995. Springer.

[60] J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials(IP): Two New Families of Asymmetric Algorithms. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'96)*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48, Springer, Berlin, Heidelberg, 1996. Springer.

[61] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *STOC '05 Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, Baltimore, MD, USA, May 22-24 2005. ACM.

[62] O. Reparaz, S. S. Roy, F. Vercauteren, and I. Verbauwhede. A Masked Ring-LWE Implementation. In *International Workshop on Cryptographic Hardware and Embedded Systems CHES 2015: Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293, pages 683–702, Springer, Berlin, Heidelberg, September 2015. Springer.

[63] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, Feburary 1978.

[64] SAFECRYPTO. Secure Architectures of Future Emerging Cryptography (SAFECrytpo). `https://www.safecrypto.eu/`, April, 2016. Online.

[65] N. Sendrier. Code-Based Cryptography: State of the Art and Perspectives. *IEEE Security & Privacy*, 15(4):44–50, August 2017.

[66] A. Shamir. Efficient Signature Schemes Based on Birational Permutations. In *Annual International Cryptology Conference CRYPTO 1993: Advances in Cryptology (CRYPTO'93)*, volume 773 of *Lecture Notes in Computer Science*, pages 1–12, Springer, Berlin, Heidelberg, 1993. Springer.

[67] P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct. 1997.

[68] A. Singh. Polynomial vs. Exponential Running Time. `http://www.superwits.com/library/design-analysis-of-algorithm/course-content-daa/polynomialvsexponentialrunningtime`, Novemeber, 2013. Online.

[69] M. Sipser. *Introduction to the Theory of Computation*. Course Technology Inc, ISBN 0-619-21764-2., 2006.

[70] S. Tang, H. Yi, J. Ding, H. Chen, and G. Chen. High-Speed Hardware Implementation of Rainbow Signature on FPGAs. In *International Workshop on Post-Quantum Cryptography PQCrypto 2011: Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 228–243, Springer, Berlin, Heidelberg, 2011. Springer.

[71] S. Tsujii, T. Itoh, A. Fujioka, K. Kurosawa, and T. Matsumoto. Public-key cryptosystem based on the difficulty of solving a system of nonlinear equations. *Electronics Letters*, 23 (11):558–560, May 21 1987.

[72] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *International Workshop on Post-Quantum Cryptography PQCrypto 2008: Post-Quantum Cryptography*, 414:883–887, Decemeber 2001.

[73] K. Wilhelm. Aspects of Hardware Methodologies for the NTRU Public Key Cryptosystem. Master's thesis, Kate Gleason College of Engineering, Rochester Institute of Technology, Rochester, NY, USA, 2008.