Analysis and Development of Ciphers Homomorphic on Addition and Multiplication

Anas Maher I. Ibrahim

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
September 2021
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Hadi Işik Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Alexander Chefranov
Supervisor

Examining Committee

1. Prof. Dr. Alexander Chefranov
2. Prof. Dr. Mehmet Ufuk Çağlayan
3. Prof. Dr. Atilla Elçi
4. Assoc. Prof. Dr. Gürcü Öz
5. Assoc. Prof. Dr. Mohammed Salamah

ABSTRACT

In this thesis, various additive and multiplicative homomorphic ciphers are analyzed and developed. New homomorphic encryption (HE) ciphers classification is proposed. The new classification of homomorphic ciphers is introduced allowing having a separate class for the newly developed herein HE cipher. It extends the previously used two criteria to five. In addition to the symmetric homomorphic scheme, HE1N, the asymmetric homomorphic schemes RSA, NTRU, RLWE-NCM-CSCM from the literature are considered.

A new ciphertext-only attack finds RSA encrypted messages as the shortest vector in a 2-dimensional lattice is designed. For RSA not to be susceptible to the attack proposed, new settings for RSA public keys are presented. NTRU and HE1N are two homomorphic cryptosystems, encrypting the message by adding to it noise and then applying modulo operation. It is found that in both of them, the modulo operation may not have an effect because the sum is less than the modulus. NTRU modulo p flaw attack against NTRU using IEEE standard parameters with non-negligible success probability is designed. To make the success probability negligible, parameter setting is recommended in the thesis. Two attacks against HE1N are designed, and new settings for HE1N parameters are recommended to mitigate these attacks.

The random congruential public-key cryptosystem (RCPKC) is developed, an NTRU variant using integers and immune against lattice basis reduction attacks (LBRA). RCPKC specifies a range from which the random numbers shall be selected to counter LBRA. Compared to NTRU, RCPKC is more efficient and it reduces energy

consumption, which allows increasing the lifetime of unattended wireless sensor networks.

Ring learning with errors (RLWE)-based cryptosystem using ciphertexts size control mechanism (CSCM), called RLWE-CSCM is developed, advancing RLWE-NCM-CSCM proposed by Brakerski and Vaikuntanathan in 2011. RLWE-CSCM is the first fully homomorphic with respect both to addition and multiplication scheme not affected by the growth of noise. The size of RLWE-CSCM ciphertext grows with each homomorphic multiplication operation. Therefore, two CSCMs are proposed in this thesis. RLWE-CSCM can be involved in a wide range of applications such as applying images filters homomorphically, homomorphic voting systems.

Keywords: homomorphic encryption scheme, addition, multiplication, known plaintext attack, ciphertext only attack, lattice basis reduction, ring learning with errors

ÖZ

Bu tezde, çeşitli toplamsal ve çarpımsal homomorfik şifreler analiz edilmiş ve geliştirilmiştir. Yeni homomorfik şifreleme (HE) şifreleme sınıflandırması önerilmiştir. Homomorfik şifrelerin yeni sınıflandırması, burada yeni geliştirilen HE şifresi için ayrı bir sınıfa izin vererek tanıtıldı. Daha önce kullanılan iki kriteri beşe kadar genişletir. Simetrik homomorfik şema HE1N'ye ek olarak, literatürden asimetrik homomorfik şemalar RSA, NTRU, RLWE-NCM-CSCM ele alınmaktadır.

Yeni bir yalnızca şifreli metin saldırısı, 2 boyutlu bir kafesteki en kısa vektör tasarlandığından RSA şifreli mesajları bulur. RSA'nın önerilen saldırıya duyarlı olmaması için, RSA ortak anahtarları için yeni ayarlar sunulur. NTRU ve HE1N, mesajı gürültü ekleyerek ve ardından modulo işlemi uygulayarak şifreleyen iki homomorfik şifreleme sistemidir. Her ikisinde de toplamın modülden daha az olması nedeniyle modulo işleminin bir etkisi olmayabileceği bulundu. IEEE standart parametreleri kullanılarak NTRU'ya karşı NTRU modulo p kusur saldırısı ihmal edilemez başarı olasılığı ile tasarlanmıştır. Başarı olasılığının ihmal edilebilir olması için tezde parametre ayarı yapılması önerilir. HE1N'ye karşı iki saldırı tasarlanmıştır ve bu saldırıları azaltmak için HE1N parametreleri için yeni ayarlar önerilir.

Rastgele uyumlu ortak anahtar şifreleme sistemi (RCPKC), tamsayılar kullanan ve kafes tabanlı azaltma saldırılarına (LBRA) karşı bağışık olan bir NTRU varyantı olarak geliştirildi. RCPKC, LBRA'ya karşı rasgele sayıların seçileceği bir aralığı belirtir. NTRU ile karşılaştırıldığında, RCPKC daha verimlidir ve enerji tüketimini azaltır, bu da gözetimsiz kablosuz sensör ağlarının ömrünün artmasına olanak tanır.

2011 yılında Brakerski ve Vaikuntanathan tarafından önerilen RLWE-NCM-CSCM'yi ilerleterek, RLWE-CSCM adı verilen şifreli metin boyut kontrol mekanizmasını (CSCM) kullanan hatalarla öğrenme (RLWE) tabanlı şifreleme sistemi geliştirildi. RLWE-CSCM, ilgili ilk tam homomorfiktir. hem toplama hem de çarpma şemasına gürültünün büyümesinden etkilenmez. RLWE-CSCM şifreli metninin boyutu, her homomorfik çarpma işlemi ile büyür. Bu nedenle, bu tezde iki CSCM önerilmiştir. RLWE-CSCM, homomorfik olarak görüntü filtreleri uygulamak, homomorfik oylama sistemleri gibi çok çeşitli uygulamalarda yer alabilir.

Anahtar Kelimeler: homomorfik şifreleme şemaları, toplama, çarpma, bilinen düz metin saldırısı, yalnızca şifreli metin saldırısı, kafes tabanlı indirgeme, ring hatalı öğrenme

DEDICATION

I dedicate the work of this thesis, firstly, to my family. My beloved wife, Nagham, your unwavering support and motivation throughout this Ph.D. project was incredible.

To my children, Maher, Sana, and Leen, you have been a gift from the beginning.

To my loving parents who constantly encouraged me and pulled me up when I would be down and under.

ACKNOWLEDGMENT

I am deeply grateful to my supervisor Professor Alexander Chefranov. It was only due to his valuable guidance, cheerful enthusiasm, and ever-friendly nature that I was able to complete this work.

I would like to express my gratitude towards my family for the encouragement which helped me in the completion of this work. My beloved and supportive wife, Nagham who is always by my side when times I needed her most and helped me a lot in making this research, and my lovable children, Maher, Sana, and Leen who served as my inspiration to pursue this work.

I would like to express my deepest gratitude to my beloved parents, who raised me to be the man I am today. You have always been a source of inspiration and give me the strength to finish this work. I also extend my gratitude to my brothers, who have always been beside me when times I need them.

I would like to acknowledge my friend Amjad for his endless support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BITRU	Binary Version of NTRU Cryptosystem Via Binary Algebra
BQTRU	Non-commutative Cryptosystem Based on Quaternion Algebras
CPKC	Congruential Public Key Cryptosystem
COA	Chosen Ciphertext-Only Attack
CRT	Chinese Remainder Theorem
CSCM	Ciphertext Size Control Mechanism
DLP	Discrete Logarithm Problem
ECC	Elliptic Curve Cryptography
ETRU	NTRU Over the Eisenstein Integers
FHE	Fully Homomorphic Encryption
GCD	Greatest Common Divisor
GLR	Gaussian Lattice Reduction
HE	Homomorphic Encryption
HE1N	Homomorphic Encryption with One Dimension
HNF	Hermite Normal Form
ICP	Ideal Coset Problem
ILTRU	An NTRU-Like Public Key Cryptosystem Over Ideal Lattices
IN-Lattice	A Lattice of Dimension Less than or Equal to N
IND-CCA2	Indistinguishability Under Adaptive Chosen Ciphertext Attack
IND-CPA	Indistinguishability Under Chosen Plaintext Attack

IPsec	Internet Protocol Security
IoT	Internet of Things
KPA	Known Plaintext Attack
L-FHE	Leveled Fully Homomorphic Encryption
LBRA	Lattice Basis Reduction Attack
LHS	Lift-Hand Side
LLL	Lenstra–Lenstra–Lovász
MITM	Meet-In-The-Middle
NCM	Noise Control Mechanism
NFS	Number Field Sieve
NIST	National Institute of Standards and Technology
NTL	Number Theory Library
NTRU	N-th Degree Truncated Polynomial Ring
OTRU	A Non-Associative Cryptosystem Based on Octonions Algebra
PHE	Partially Homomorphic Encryption
PKC	Public-key Cryptosystems
RCPKC	Random CPKC
QTRU	NTRU Variant Based on Quaternion Algebra
RHS	Right-Hand Side
RSA	Rivest–Shamir–Adleman
RLWE	Ring Learning with Errors
SHE	Somewhat Homomorphic Encryption
SVP	Shortest Vector Problem
w.r.t	With Respect to

WSN	Wireless Sensor Network
X.509	Standard Defining the Format of Public-Key Certificates

Chapter 1

INTRODUCTION

The Internet of Things (IoT) is a network paradigm enabling an enormous amount of devices and data to be shared, processed, and stored (Z. Han et al., 2018; Margaret Amala & Gnana Jayanthi, 2020). On the other hand, cloud computing is a cost-effective approach enabling customers to benefit from high-performance computing and virtually unlimited storage resource (Ramesh & Govindarasu, 2020; Sinchana & Savithramma, 2020). CloudIoT has recently emerged as a paradigm leveraging IoT and cloud computing technologies (Benabbes & Hemam, 2019). Different applications such as wearables (e.g., smartwatches, fitness trackers), self-driving cars, healthcare (Shah & Bhat, 2020), smart grid (Rabie et al., 2021), smart cities (Kelaidonis et al., 2017), and surveillance systems benefit from CloudIot. Due to the enormous amount of sensitive and personal data exchanged, processed, and stored in this environment, the importance of preserving data privacy (during transmitting or storing) becomes critical.

Public key cryptosystems (PKC), such as RSA (R. L. Rivest, Shamir, and Adleman 1978) and ECC (Stallings 2017, 330), are widely used nowadays in CloudIoT for verifying node identities to prevent weak authentication (Yakubu et al. 2019, 226; Pandey, Pandey, and Kumar 2020, 321). RSA is used in the Internet Key Exchange protocol that is designed specifically for use with Internet Protocol Security (IPsec) (Barker & Dang, n.d., p. 24) to provide peer authentication. RSA is also widely used

in X.509, the standard defining the format of public-key certificates (Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002). PKC schemes provide security to the shared data in the environment, but performing operations over encrypted data, such as query databases, requires data to be decrypted first, and thus, compromises the privacy of sensitive and personal data stored on an untrusted third-party cloud server.

Homomorphic encryption (HE) scheme is a form of encryption that allows performing computations over encrypted data without the decryption keys, i.e., given encryptions, $E(m_1), E(m_2)$ of plaintexts m_1, m_2, then, $E(m_1) \diamond E(m_2) = E(m_1 \diamond m_2)$. For example, RSA cryptosystem (R. L. Rivest, Shamir, and Adleman 1978) has the property of multiplicative homomorphism, i.e., a party having RSA public key $pk = (e, N)$ and ciphertexts $\{c_i = m_i^e \ mod \ N\}$, can compute $\prod_i c_i = (\prod_i m_i)^e \ mod \ N$, a ciphertext that encrypts the product of the original plaintexts. Many applications involve RSA multiplicative homomorphism feature such as secure image sharing (Islam et al., 2011), and homomorphic signatures(R. Johnson et al. 2002; Freeman 2012). HE schemes allow CloudIoT users to benefit from cloud services without compromising data confidentiality or their privacy (Ramesh and Govindarasu 2020).

Some HE schemes can only support a limited number of allowed homomorphic operation types, such as RSA supports only the multiplication operation. Due to increasing the number of applied homomorphic operations, the noise used to mask the message increased, and thus, decryption fails if the value of the noise exceeds some predefined threshold. Therefore, such HE schemes can support a limited number of times to apply homomorphic operations. HE schemes can have an increase in the ciphertext size (number of components) with each homomorphic operation. Several

classifications have been proposed (Acar et al., 2018; Domingo-Ferrer et al., 2019; Feng et al., 2020; Martins et al., 2017; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020), these classifications depend on a single criterion, that is the number of supported homomorphic operations type (Domingo-Ferrer et al., 2019; Feng et al., 2020; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020), or the underlying hard problem of the HE scheme (Acar et al., 2018; Martins et al., 2017). The use of few classification criteria leads to group dissimilar HE schemes in one class, for example, using the number of supported homomorphic operation types as a classification criterion (Domingo-Ferrer et al., 2019; Feng et al., 2020; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020) leads to group HE schemes having increasing ciphertext size (number of components) such as (Brakerski et al., 2013; Brakerski & Vaikuntanathan, 2011b, 2011a) with HE schemes that don't have increasing ciphertext size such as (Gentry, 2009b; Hoffstein et al., 1998). To overcome this issue, a new classification using five criteria is proposed in Chapter 2. Using this classification, known HE schemes are grouped into eight different classes (see Table 2.1 in Chapter 2).

Due to the importance of RSA in preserving the privacy during transmitting or/and storing the data, analysis of RSA security is conducted in Sections 2.23.1, 3.2. Design for a cipher-text-only attack (COA) against RSA encrypted message using the lattice basis reduction algorithm LLL (Ibrahim, Chefranov, and Hamamreh 2021) is performed in Sections 3.3-3.5.

The computational complexity for RSA is high due to modular exponentiation operations in the encryption/decryption process, which make such PKC schemes unsuitable for limited resources devices (Sethi et al. 2021), especially when using large

bit key size, such as 2048-bit for RSA as suggested by NIST in 2020 (Barker & Dang, n.d., p. 54), to meet the minimal security strength of 112-bit. Compared to RSA, elliptic curve cryptography (ECC) offers equal security for a smaller key size (Stallings 2017). ECC is included in the IEEE P1363 Standard for Public-Key Cryptography ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009). A transition to post-quantum algorithms is needed as cryptanalytic algorithms, such as Shor's factorization algorithm, are developed to defeat the security provided by currently approved asymmetric algorithms(Barker, n.d., p. 27). NTRU (Jeffrey Hoffstein, Pipher, and Silverman 1998) cryptosystem, standardized as IEEE P1363.1("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009), is faster than RSA and ECC (Hermans, Vercauteren, and Preneel 2010), is announced as one of seven candidate algorithms in the third-round finalists of NIST Post Quantum Cryptosystem Standardization Process ("PQC Third Round Candidate Announcement | CSRC" n.d.). NTRU, from Class 3, is homomorphic with respect to (w.r.t) two operations, multiplication and addition, therefore serves as a base to many homomorphic cryptosystems (Yarkln Doröz and Sunar 2020; Yarkın Doröz et al. 2018; López-Alt, Tromer, and Vaikuntanathan 2012). Analysis of NTRU security is performed, and design of a new attack based on a flaw that allows for some parameters to compromise the encrypted message just by applying modulo p operation to the ciphertext, where p is a public parameter is performed in Section 4.1. On the other hand, NTRU is susceptible to LBRA using LLL (Jeffrey Hoffstein, Pipher, and Silverman 1998). Yang et. al. (Z. Yang et al. 2018) proposed a lower dimension lattice (w.r.t. the normal $2N$-dimensional NTRU lattice (Jeffrey Hoffstein, Pipher, and Silverman 1998, 273)) attack, called $IN - Lattice$ attack. Experiments analysis of

attack is conducted in Section 4.2, we found that $IN - Lattice$ is not so efficient as claimed in the paper. Thus, NTRU resists lattice basis reduction attacks by increasing the polynomial degree N, which leads to increase of the computational complexity of encryption/ decryption processes.

In Section 6.1, CPKC (Jeffrey Hoffstein, Pipher, and Silverman 2014a) is analyzed. CPKC proposed by NTRU authors as a toy model prone to lattice attacks and uses polynomials of zero degree, that is integers modulo $q >> 1$, thus, it outperforms NTRU. In Section 6.2, we developed NTRU-like RCPKC (Ibrahim et al. 2020), a random congruential public key cryptosystem using integers, based on CPKC, immune against lattice-based attacks, and faster than NTRU and its variants.

HE1N cryptosystem (Dyer, Dyer, and Xu 2019), is homomorphic w.r.t two operations, and proved to be IND-CPA secure. HE1N, from Class 4, is recently proposed in (Dyer, Dyer, and Xu 2019) as a practical solution to the problem of privacy in the cloud providing homomorphism with respect to two operations, and efficient encryption as working on integers and performing simple modular arithmetic operations. Analysis of HE1N security is conducted, and the design of a new ciphertext-only attack, and known-plaintext attack (KPA) against HE1N is performed in Chapter 5.

Due to increasing the noise used to mask the plaintext with increasing the number of homomorphic operations, HE1N allows computing function with a limited number of additions and multiplications on ciphertexts, such schemes are called leveled fully homomorphic encryption (L-FHE) schemes. L-FHE schemes by itself are suitable for some applications (Brakerski, 2019). In (Gentry, 2009a), Gentry proposed the first HE scheme that can compute arbitrary functions of any number of additions and

multiplications on ciphertexts, such scheme is called fully homomorphic encryption (FHE). In his proposal (Gentry, 2009a, 2009b) from Class 7. Gentry uses "*Bootstrapping*" to convert the L-FHE scheme to the FHE scheme.

After Gentry's FHE proposal in (Gentry 2009a; 2009b), many FHE schemes are proposed using different objects. All these FHE schemes are constructed from L-FHE schemes by using a noise control mechanism (NCM) such as bootstrapping (Gentry 2009a), or modulus switching (Brakerski et al., 2014). FHE scheme (Brakerski & Vaikuntanathan, 2011b), from Class 6, using ring learning with errors (RLWE) is L-FHE as it is affected with noise, and the ciphertext size (number of components) growth with the growth of the number of multiplication operations. Thus, hereafter, we call it RLWE-NCM-CSCM, as it is needs NCM and ciphertext size control mechanism (CSCM). In Chapter 7, RLWE-CSCM is proposed, the first FHE scheme not affected with noise growth by construction, thus, no NCM is needed. RLWE-CSCM, is RLWE based and its ciphertext size increases with the growth of the number of multiplications. Thus, we propose two different CSCM to control the growth of ciphertext size.

The rest of the dissertation is organized as follows. Chapter 2, provides a new classification for HE schemes, and reviews the state of the art. In Chapter 3, of RSA security is analyzed, and a new ciphertext-only attack against RSA encrypted message using lattice-based reduction is proposed. In Chapter 4, NTRU PKC security is analyzed, a new attack against RSA message is designed, and experimental analysis of the recently published attack, $IN-Lattice$ attack (Z. Yang et al. 2018) is performed. Analysis of HE1N is security and development of several attacks again HE1N is performed in Chapter 5. Development of RCPKC, NTRU-like cryptosystem,

faster and more secure than NTRU is executed in Chapter 6. Development of RLWE-CSCM, the first FHE scheme not affected by the growth of noise by structure, is performed in Chapter 7. Chapter 8 concludes the dissertation. Thesis contributions can be summarized as follows:

1- New HE schemes classification is proposed extending the previously used criteria from two to five, and increases the number of classes from four to at least 32.
2- A new COA attack against RSA private Key/ message using LBRA with LLL algorithm (Ibrahim, Chefranov, and Hamamreh 2021) is designed.
3- A new attack against NTRU message, NTRU Modulo p Flaw attack (Chefranov & Ibrahim, 2016; Ibrahim & Chefranov, 2016), is designed.
4- Experimental experiments are conducted to verify the efficiency of IN-Lattice attack (Z. Yang et al., 2018b) against NTRU keys (Easttom et al., 2020).
5- RCPKC (Ibrahim et al., 2020), an efficient and secure variant of NTRU is developed.
6- Several attacks are designed against HE1N (Dyer et al., 2019).
7- RLWE-CSCM, the first fully homomorphic cryptosystem without noise control mechanism, advancing RLWE-NCM-CSCM by Brakerski et. al. (CRYPTO 2011).

Chapter 2

ANALYSIS OF HOMOMORPHIC SCHEMES AND LATTICE ATTACKS ON THEM

In this chapter, an analysis of HE schemes is presented, a new classification for HE schemes is proposed, state of the art is reviewed and problems addressed by this dissertation are stated. The rest of the chapter is as follows; in Section 2.1 an analysis of HE schemes is provided, and a new classification for HE schemes is proposed. In Section 2.2 reviews state of the art. Section 2.8 defines the problems addressed in this dissertation.

2.1 Homomorphic Encryption Schemes and Their Classification

HE schemes allow performing computations over its ciphertexts (Pulido-Gaytan et al. 2021), where the computations are represented as arithmetic functions (circuits) (Brakerski, 2019), according to Definition 2.1 below:

Definition 2.1: (Gentry, 2009b). Encryption scheme $\mathcal{E}$ = ($\text{KeyGen}_{\mathcal{E}}$, $\text{Encrypt}_{\mathcal{E}}$, $\text{Decrypt}_{\mathcal{E}}$, $\text{Evaluate}_{\mathcal{E}}$) is homomorphic if,

$$c = \text{Evaluate}_{\mathcal{E}}(\text{pk}, F, C) \Rightarrow F(m_0, .., m_{t-1}) = \text{Decrypt}_{\mathcal{E}}(\text{sk}, \text{c}), \quad (2.1)$$

where $C = (c_0, \ldots, c_{t-1})$ is a tuple of ciphertexts encrypting plaintexts $(m_0, \ldots, m_{t-1})$, using the public key pk (with respective secret key sk), in case of asymmetric HE, and sk only for symmetric HE, a function $F \in \mathcal{F}_{\mathcal{E}}$, a set containing all functions that can be homomorphically evaluated by the scheme $\mathcal{E}$ (for them (2.1) holds). It is most

common to consider the arithmetic circuit model to represent F. The algorithms $\text{KeyGen}_{\mathcal{E}}$, $\text{Encrypt}_{\mathcal{E}}$, $\text{Decrypt}_{\mathcal{E}}$ satisfy the following:

- $\text{KeyGen}_{\mathcal{E}}$ outputs a public-key (pk) used for encryption, and the corresponding secret-key (sk) used for decryption in case of asymmetric HE, and sk only for symmetric HE;
- $\text{Encrypt}_{\mathcal{E}}$ encrypts a plaintext using the public-key pk for asymmetric, or secret key, sk, for symmetric HE;
- $\text{Decrypt}_{\mathcal{E}}$ decrypts the ciphertext, using the corresponding secret-key sk;
- The computational complexity of $\text{KeyGen}_{\mathcal{E}}$, $\text{Encrypt}_{\mathcal{E}}$, $\text{Decrypt}_{\mathcal{E}}$, must be polynomial in security parameter λ;
- The computational complexity of $\text{Evaluate}_{\mathcal{E}}$, must be polynomial in the depth of δ, the arithmetic circuit representing F.

Definition of the circuit depth follows.

Definition 2.2: (Sipser 2013, 382) The depth (number of levels) of a circuit is the length (number of gates) of the longest path from the input gates to the output gates.

For example, arithmetic circuit calculating $y = (x_1 + x_2) \times x_3 + x_4 \times x_5$ shown in Figure 2.1 has depth equals to 3.

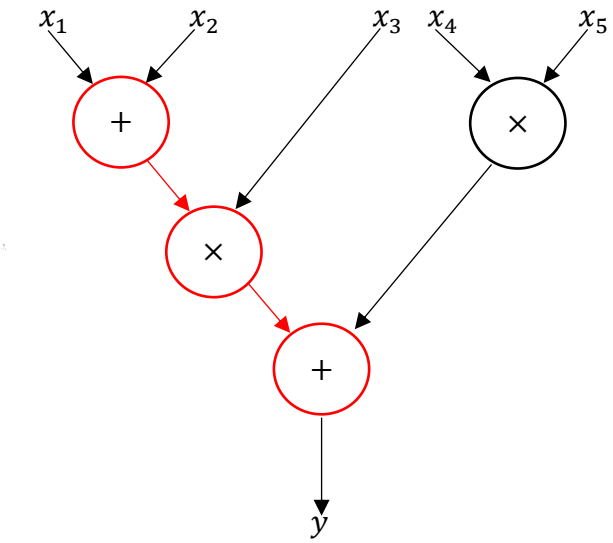

Figure 2.1: Depth 3 arithmetic circuit calculating $y = (x_1 + x_2) \times x_3 + x_4 \times x_5$, with the longest path shown in red.

HE schemes such as RSA (R. L. Rivest, Shamir, and Adleman 1978), El-Gamal (ElGamal 1985), can only evaluate homomorphically function F in (2.1), that is composed of one type of arithmetic operations, and with no constraints on the depth of the arithmetic circuit represents F, such schemes are called partially homomorphic encryption (PHE) schemes (Acar et al., 2018; Martins et al., 2017; Shrestha & Kim, 2019), according to Definition 2.3,

Definition 2.3: (Acar et al., 2018, p. 3) "*Partially homomorphic encryption (PHE) scheme allows only one type of operation with an unlimited number of times (i.e., no bound on the number of usages)*".

In other words, PHE scheme supports one type of operations, with an unlimited number of times this type can be applied. On the other hand, L-FHE schemes such as NTRU (Jeffrey Hoffstein, Pipher, and Silverman 1998) and HE1N (Dyer, Dyer, and Xu 2019), can evaluate function F in (2.1), composed of more than one type of

arithmetic operations, but with limited depth of the arithmetic circuit represents F, according to Definition 2.4 below,

Definition 2.4: (Brakerski, 2019, p. 549). L-FHE scheme is the HE scheme $\mathcal{E}^{(d)}$ that allows, for some $d \in Z^+$, evaluation of depth-d circuits, i.e., $\mathcal{F}_{\mathcal{E}^{(d)}}$, is a set of functions represented by circuits of depth bounded by d. The parameters of the scheme are allowed to grow polynomially with d.

In some works (Sun et al. 2017; Youn, Jho, and Chang 2016; Boulemtafes et al. 2021), the term "somewhat homomorphic encryption" (SHE) is used to indicate a scheme with homomorphic capabilities against a restricted class of functions (depth bounded). In this dissertation, the terms L-FHE and SHE are used interchangeably.

In L-FHE schemes, increasing the number of applied homomorphic operations, leads to the growth of parameters used to mask the plaintext (noise), and thus, decryption fails if the value of the noise exceeds some predefined threshold. In addition to the growth of noise, some L-FHE schemes have also a growth of ciphertext size (number of components). Gentry in (Gentry 2009a; 2009b) proposed Bootstrapping, to control the growth of noise. Using Bootstrapping, Gentry managed to remove the constraints on the circuit depth of the evaluated function, and thus converted an L-FHE scheme into an FHE scheme defined by Definition 2.5,

Definition 2.5: (Brakerski, 2019, p. 549). A homomorphic encryption scheme $\mathcal{E}$ is FHE if $\mathcal{F}_{\mathcal{E}}$ is the set of all functions (or at least the set of all efficiently computable functions).

All known FHE schemes, are following Gentry's method of converting the L-FHE to FHE using a NCM, or both of NCM and CSCM in the case of the L-FHE scheme has

a growth of both of ciphertexts size and the noise masks the message.. Several classification attempts are made for HE schemes (Acar et al., 2018; Domingo-Ferrer et al., 2019; Feng et al., 2020; Martins et al., 2017; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020). In (Domingo-Ferrer et al., 2019; Feng et al., 2020; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020), HE schemes are classified into PHE, SHE, FHE w.r.t two criteria: operation type, and the allowed number of homomorphic operations. Such classification criteria don't show the difference between L-FHE schemes need NCM only, and the L-FHE schemes need CSCM only (to be proposed in Chapter 7), or need both of them. HE schemes differ in:

1- the number of homomorphic operation types: while PHE supports one operation type, L-FHE supports more than the operation type.
2- the need for NCM: while PHE supports performing an unlimited number of times of a specific operation type, L-FHE supports a limited number of times of several operation types.
3- the need for CSCM; performing homomorphic operations leads to increase the size of some L-FHE's ciphertext (number of components) such as (Brakerski et al., 2014; Brakerski & Vaikuntanathan, 2011a).
4- the underlying hard problem(s): HE with similar homomorphic features can be differ in the underlying hard problem(s). For example, both of RSA (R. L. Rivest, Shamir, and Adleman 1978) and El-Gamal (ElGamal 1985) support one type of operation, don't need NCM or CSCM, and they only differ in the underlying hard problem (see Table 2.1).
5- and the number of keys used: HE schemes can be symmetric or asymmetric.

Therefore, new classification is proposed. It extends the previously used criteria (Acar et al., 2018) (operation type, and the allowed number of homomorphic operations) to five:

1- number of homomorphic operation types;
2- the need for NCM;
3- the need for CSCM;
4- the underlying hard problem(s);
5- and the number of keys used.

In the new criteria list, the previously used criterion, allowed number of homomorphic operations, is replaced by: the need for NCM, and the need for CSCM, so the new classification categorizes L-FHE schemes according to the control mechanism used.

The proposed classification of known HE schmes is presented in Table 2.1. From Table 2.1, we can notice that:

1- PHE schemes (Classes 1, 2) are asymmetric encryption schemes, differing only in the underlying hard problem. Therefore, in Section 2.2 we consider one scheme as representative of Classes 1, 2 that is RSA.
2- All HE schemes support more than one homomorphic operation type, Classes 3-8, either need NCM only, or need both NCM and CSCM.

Section 2.2 reviews HE schemes from Classes 1-9, Table 2.1.

Table 2.1: Classification of homomorphic schemes (col. 2) based on five criteria (cols. 3-7).

1	2	3	4	5	6	7
class number	**Particular instances of homomorphic schemes**	**Number. of homomorphic arithmetic operations types (1, 2)**	**Needs NCM (Yes, No)**	**Needs Ciphertext Size Control Mechanism (Yes, No)**	**Underlying hard problem (Set of problems defining methods resistance)**	**Number of keys (Symmetric-1, Asymmetric-2)**
1	RSA(R. L. Rivest, Shamir, and Adleman 1978)	1	No	No	Discrete logarithm, integer factorization	2
2	El-Gamal(ElGamal 1985)	1	No	No	Discrete logarithm	2
3	NTRU(Jeffrey Hoffstein, Pipher, and Silverman 1998), López-Alt et al. (López-Alt, Tromer, and Vaikuntanathan 2012)	2	Yes	No	Shortest Vector Problem in a Lattice(Jeffrey Hoffstein, Pipher, and Silverman 1998, 273)	2
4	Gentry(Gentry 2010), Van Dijk et al. (van Dijk et al. 2010), HE1N(Dyer, Dyer, and Xu 2019)	2	Yes	No	Approximate GCD (Gentry 2010, 101)	1
5	Coron et. al. (Coron, Naccache, and Tibouchi 2012; Coron et al. 2011), Yang et. al. (H. M. Yang et al. 2012), Cheon et. al. (Cheon et al. 2013), Chen et. al.(Chen, Ben, and Huang 2014), Ramaiah and Kumari (Ramaiah and Kumari 2012), Nuida and Kurosawa (Nuida and Kurosawa 2015)	2	Yes	No	Approximate GCD	2

6	Brakerski and Vaikuntanathan (Brakerski & Vaikuntanathan, 2011a, 2011b), Brakerski et. al. (Brakerski et al., 2014, 2013)	2	Yes	Yes	Learning with errors (LWE)	2
7	Gentry(Gentry 2009a)	2	Yes	No	Ideal Coset Problem (ICP) (Gentry 2009a, sec. 3.2)	2
8	Smart and Vercauteren (Smart and Vercauteren 2010)	2	Yes	No	Small Principal Ideal Problem, Polynomial Coset Problem (Smart and Vercauteren 2010, 429,431)	2

2.2 Review of Clasps 1 HE Schemes (RSA PKC)

In this section, RSA from Class 1 is reviewed.

2.2.1 Review of RSA PKC

A message, $m \in Z_N$, is encrypted using,

$$c = m^e mod\ N, \tag{2.2}$$

where $N = p \cdot q$, p and q are two different prime numbers, and the encryption exponent, *e*, is chosen according to,

$$\gcd\big(e, (p-1)(q-1)\big) = 1 \tag{2.3}$$

The message, $m \in \mathbb{Z}_N$, is retrieved by decryption of the ciphertext, c, from (2.2) as follows,

$$m = c^d mod\ N, \tag{2.4}$$

where the decryption exponent, *d*, is the multiplicative inverse of *e* satisfying,

$$e \cdot d\ mod\ (p-1)(q-1) = 1. \tag{2.5}$$

The public key is (N, e), and the private key is (N, d).

An example of 40-bit RSA encryption/ decryption process is provided in Example A.1, Appendix A. Section 2.2.2 shows RSA homomorphism, Section 2.2.3 reviews lattice-based attacks on RSA, and in Chapter 3, a new ciphertext-only attack (COA) (Ibrahim, Chefranov, and Hamamreh 2021) against RSA encrypted messages is proposed.

2.2.2 Homomorphism of RSA with Respect to Multiplication

RSA can only evaluate homomorphically unlimited number of multiplication operations on ciphertexts as shown below:

From (2.2)

$$c_{(i)} = m_{(i)}^e\ mod\ N, i = \{1, \dots, n\}$$

Thus,

$$C_{mult} = c_{(1)} \cdot c_{(2)} \cdot \ldots \cdot c_{(n)} = m_{(1)}^{e} \cdot m_{(2)}^{e} \cdot \ldots \cdot m_{(n)}^{e} \; mod \; N = \left(\prod_{i=1}^{n} m_{(i)} \right)^{e} \; mod \; N.$$

Thus, C_{mult} is the encryption of $\prod_{i=1}^{n} m_{(i)}$, and computed without the need to decrypt the ciphertexts $c_{(1)}, \ldots, c_{(n)}$. Therefore, RSA is homomorphic w.r.t to one type of operations that is multiplication. RSA encryption (2.2) does not use any noise to mask the plaintext. Thus, RSA does not need NCM or CSCM.

In Chapter 3 a new ciphertext-only attack on RSA using lattices basis reduction is proposed. Therefore, Section 2.2.3 reviews lattice-based attacks on RSA. Appendix B provides a brief introduction to lattices and lattice basis reduction algorithms.

2.2.3 Review of Lattice-Based Attacks on RSA

LLL algorithm (A. K. Lenstra, Lenstra, and Lovász 1982) of lattice basis reduction is used for COA on RSA (Boneh et al., 1998; Coppersmith, 1996b, 1996a; Hastad, 1988, 1986; Takayasu & Kunihiro, 2019, 2014) and NTRU (Jeff Hoffstein et al. 2010; Jeffrey Hoffstein, Pipher, and Silverman 1998; Kirchner and Fouque 2017; Z. Yang et al. 2018). Most of the attacks require either message broadcasting, or prior knowledge of a part of a message/private key. And the problem of attacking RSA is considered as a problem of solving SVP in a lattice dimension of which grows with the growth of the encryption exponent, *e*. LLL algorithm computational complexity exponentially depends on the lattice dimension (Jeffrey Hoffstein, Pipher, and Silverman 2014b, 428), equation (7.49), (Jeffrey Hoffstein, Silverman, and Whyte 1999, 6), Table 1, and, hence, it solves SVP efficiently for low-dimensional lattices but the solution is infeasible for lattices with a dimension greater than 400 (“IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices,”

2009) to meet minimum security level of 112-bit. That is why, attacks on RSA using LLL assume low encryption exponent value (May 2010). next, the attacks on RSA private key and plaintext message are reviewed.

2.2.3.1 Lattice-Based Attacks Against RSA Private Key

In (Coppersmith, 1996a), prime factors of $N = p \cdot q$, used as the modulus value in RSA encryption (2.2) with the public key, e, and decryption (2.4) with the private key, d, are found as roots of a bivariate polynomial constructed using high order $\left(\frac{1}{4} + \varepsilon\right) \log_2 N$ bits of p, $\varepsilon > 2/\log_2 N$ (high order bits of q are known by division of N by p). LLL lattice basis reduction algorithm is used for dimension $r = 2k + 1, k > 1/(4\varepsilon)$. In (Boneh & Durfee, 2000, 1999), LLL method is used to disclose the private key, $d < N^{\delta}, \delta = 0.292$, that extends the attack applicability compared to attack (Wiener 1990) assuming $\delta = 0.25$. In (de Weger, 2002), two parameters define the attack applicability: δ, and β such that $\Delta = |p - q| = N^{\beta}$. In (de Weger, 2002, fig. 1), specifies that known attacks on RSA are not applicable for $\beta \in [0.5, 1]$, and mainly not applicable for $\delta \in [0.5, 1], \beta \in [0.25, 0.5]$. Using β, the attack extends the applicability of (Boneh & Durfee, 2000, 1999) attack up to $\delta \to 1$ for $\beta \to 0.25 + \varepsilon, \varepsilon \to 0$. In (Takayasu and Kunihiro 2014; 2019; Boneh, Durfee, and Franke 1998), LLL algorithm is used to disclose secret RSA exponent provided that part of it (least- or most-significant bits) are known. (Takayasu and Kunihiro 2019, fig. 1; 2014, fig. 2) show that δ can be extended to 0.57 and 0.37 for the use of most- and least-significant bits, respectively.

2.2.3.2 Lattice-Based COA Against RSA Messages

In (Coppersmith, 1996b), an encrypted RSA message is disclosed as a root of a univariate polynomial of low order, e. Exponent considered in the paper is $e = 3$ resulting in the polynomial of order, $k = e = 3$. The message, m, to be found shall be

rather small: $|m| < N^{\frac{1}{k}-\varepsilon}$, $\varepsilon = \frac{1}{\log N}, k < \log N$, (see (Hastad 1986, sec. 2)). Respective lattice size is,

$$Size = 2h \cdot k - k \geq 2 \cdot \frac{1}{k\varepsilon} k - k = 2\log_2 N - k > 0, \tag{2.6}$$

where h is such that $h \cdot k \geq 7$ and $h - 1 \geq (hk - 1)(\frac{1}{k} - \varepsilon)$. It is known from NTRU security requirements ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009) that if the size of a lattice meets, $Size \geq 400$, then LLL attack is unfeasible. Thus, from (2.6), it follows that already for 512-bit RSA the attack is not feasible because $\log_2 N = 512$, $k < 512$, and, hence, $Size \geq 1024 - k \geq 512$. Note that in (Coppersmith, 1996b), estimates of RSA parameters, such that the proposed attack is feasible, are not defined. In (Hastad 1986), Hastad showed that the message, m, can be revealed in polynomial time when it is encrypted with several public keys, (e_i, N_i), each having the same public exponent, e, and different moduli values, $N_i, i = 1, \cdots, k$, expected to be mutually relatively prime, and meeting (2.7):

$$m < \min_{i=1,\cdots,k} N_i, \tag{2.7}$$

$$k > \frac{e(e+1)}{2}, \tag{2.8}$$

$$N > n^{\frac{e(e+1)}{2}} (k + e + 1)^{\frac{(k+e+1)}{2}} 2^{\frac{(k+e+1)^2}{2}} (e+1)^{e+1}, \tag{2.9}$$

$$N = \prod_{i=1}^{k} N_i, \tag{2.10}$$

(see (Hastad 1986, 405)). Method (Hastad 1988) is practically the same as in (Hastad 1986) with slightly different lattice constructed, and, thus, slightly differing from (2.9) (see (Hastad 1988, 338)), and is also applied to a broadcasted message. The

broadcasted message, m, is revealed by applying LLL algorithm to the lattice defined using coefficients of the polynomials resulting from the message encryption using different moduli. Also, Chinese remainder theorem (CRT) is used.

2.2.3.3 Non-Lattice-Based Attacks Against RSA Private Keys and Messages

RSA secret key can be disclosed if the integer modulus, N, is factorized. Methods of integer factorization are reviewed in (Rabah 2006; Alimoradi and Arkian 2016), and application of one of them, number field sieve (NFS), in (Kleinjung et al. 2010) in December 2009, resulted in factoring 768-bit RSA modulus, RSA-768. RSA moduli RSA-240 and RSA-250 with 795 and 829 bits were factored by NFS on December 2, 2019, and February 28, 2020, ("RSA Factoring Challenge - Wikipedia" n.d.) which took 4000 and 2700 core-years of Intel Xeon Gold 6130 CPUs as a reference (2.1GHz), respectively (*Listserv - Nmbrthry Archives*, n.d.),(*[Cado-Nfs-Discuss] Factorization of RSA-250*, n.d.).

In (Bunder et al. 2017), a method of factoring RSA modulus, $N = pq, q < p < 2q$, in time polynomial in $\log N$ is proposed under assumption that an encryption exponent, e, meets $e \cdot x - (p^2-1)(q^2-1)y = z,\ gcd(y,x) = 1,\ z \neq 0,\ x \cdot y < 2N - 4\sqrt{2}N^{\frac{3}{4}},\ |z| < (p-q)N^{\frac{1}{4}}y,\ e < (p^2-1)(q^2-1)$. In (Wiener 1990), continued fractions are used for d disclosure with $\delta \leq 0.25$. In (Wu et al. 2014), the applicability of the attack (Wiener 1990) is extended to $d \leq N^{\delta} \cdot 2^r,\ r \leq 7$. In(Abd Ghafar et al., 2020), a method of N factorization is proposed applicable when $|N^{0.5} - p^{0.5} \cdot q^{0.5}|$ is sufficiently small (less than 2^{112} as explained in (Abd Ghafar et al., 2020, p. 4)).

Super-encryption (successive encryption of the ciphertexts) is proposed in (Simmons 1977; Berkovits 1982). However, in (Jamnig 1988; Ronald L. Rivest 1978), it is shown

that the probability of success is about 10^{-90} for the parameters proposed for RSA in (Rivest et al., 1978) because $p-1, q-1$ shall have large prime factors, and similar for them as well. In (Bleichenbacher, 1997), Bleichenbacher defines that plaintexts m_i are related if $m_i = f_i(m)$ for some known polynomials f_i and shows that having l RSA public keys $(e_1, N_1), \cdots, (e_l, N_l), N = N_1 N_2 \cdots N_l$ and $c_i = f_i(m)^{e_i} \bmod N_i$ for $i = 1, \cdots, l$, the plaintext m can be computed in time polynomial in $\log N$ using Coppersmith's algorithm (Coppersmith, 1996b, p. 156). A method of the broadcasted message disclosure is proposed based on the use of the CRT allowing reducing the number of modular equations to a single equation and then finding e-th order root over integers, in the simplest case of broadcasting one and the same message, (Bleichenbacher, 1997, p. 241), or a univariate polynomial root finding using Coppersmith method (Coppersmith, 1996b) for broadcasting related messages based on a small message. The paper considers messages, m_i, related to the base message, m, by an affine transformation, $m_i = \alpha_i \cdot m + \beta_i \, mod \, n_i$ (Bleichenbacher, 1997, p. 242), whereas in Coppersmith method only translation transformation is expected to be used: $m' = m + t$ (Coppersmith, 1996b, p. 161). In (Delaurentis 1984), DeLaurentis considered two cases. In Case 1, a probabilistic algorithm is proposed that allows factoring modulus, $N = p \cdot q$, using information on the public-private key pair of the attacker (insider) but not of the other users, neither public, nor private keys, within an average number of runs at most 2. In Case 2, without factoring of N, an own encryption-decryption key pair, as well as an encryption key of another valid user are used to disclose an equivalent for the private key of another user that may be used to disclose his messages and to forge his signature. Simmons (Belhaj & Kahla, 2013) considers one message encrypted by two different encryption keys resulting in two ciphertexts of one and the same message. If the encryption keys are co-prime, their

mutual inverses may be found and used for the message disclosing. In (Arjen K. Lenstra and Verheul 2000, 265), Lenstra et. al., clarified that selecting values of public exponent e, such as 3 and 17 can no longer be recommended, but commonly used values such as $2^{16} + 1 = 65537$ still seems to be fine. Lenstra et. al. (Arjen K. Lenstra and Verheul 2000, 265) asserted that if one prefers to stay on the safe side, an odd 32-bit or 64-bit of public exponent at random may be chosen.

If a known plaintext-ciphertext pair, (P, C) is known, discrete logarithm problem (DLP) solution can be used to disclose the private key as $d = log_{C,N} P$. DLP computational complexity is of the order of that of integer factorization and in parallel with factorization respective DLP solving is reported in (*[Cado-Nfs-Discuss] Factorization of RSA-250*, n.d.; *Listserv - Nmbrthry Archives*, n.d.).

In (Coppersmith et al., 1996), a method for recovering RSA messages is proposed for rather large encryption exponent such as, $e = 2^{16} + 1$. The method assumes that two plain messages are encrypted with the same encryption exponent, e, and modulus, N, and one of the messages, m_2, is related with another one, m_1, by an affine transformation, $m_2 = a \cdot m_1 + b$, and two respective ciphertexts are known, c_1, c_2. The message, m_1, is found as a root of a polynomial which is the greatest common divisor (GCD) of two univariate polynomials modulo N, $p_1(m_1) = m_1^e - c_1, p_2(m_1) = (a \cdot m_1 + b)^e - c_2$. The GCD is obtained using Euclid's algorithm. The method is generalized for the cases of $m_2 = p(m_1)$, where $p(\quad)$ is a polynomial, and for multiple messages polynomially related, $p(m_1, \cdots, m_k) = 0$. As far as all the related messages, $m_2, \cdots, m_k$ depend on the single message, m_1, this mode of operation can be considered as "broadcasting" of the message m_1 and its dependent messages,

$m_2, \cdots, m_k$ encrypted each with its own encryption exponent. The maximal encryption exponent mentioned in the paper is $e = 2^{16} + 1$. Figure 2.2 shows our implementation of the attack (Coppersmith et al., 1996) using Maple 2016 on Intel i7-7700 CPU 3.60 GHz, 8GB RAM. The message, $m = 2$, is recovered as the root of the GCD of two univariate polynomials, $p_1 = x^e - c_1 \bmod N$ and $p_2 = (x+1)^e - c_2 \bmod N$, where $N = p \cdot q = (2^{20} + 7) \cdot (220 + 13)$, $e = 2^{16} + 1$, $c_1 = m^e \, mod \, N$, $c_2 = (m + 1)^e \, mod \, N$. The GCD is found nearly in 6 minutes.

$m := 2$

$$2 \quad (1)$$

$p := 2^{20} + 7$

$$1048583 \quad (2)$$

$q := 2^{20} + 13$

$$1048589 \quad (3)$$

$N := p \cdot q$

$$1099532599387 \quad (4)$$

$e := 2^{16} + 1$

$$65537 \quad (5)$$

$c1 := m^e \bmod N$

$$283976582449 \quad (6)$$

$c2 := (m+1)^e \bmod N$

$$217618377630 \quad (7)$$

$p1 := (x^e - c1) \bmod N$

$$x^{65537} + 815556016938 \quad (8)$$

$p2 := ((x+1)^e - c2) \bmod N$

$$(x+1)^{65537} + 881914221757 \quad (9)$$

$s := Gcd(p1, p2) \bmod N$

$$x + 1099532599385 \quad (10)$$

$time(Gcd(p1, p2) \bmod N)$

$$343.953 \quad (11)$$

$-1099532599385 \bmod N$

$$2 \quad (12)$$

Figure 2.2: Maple code implementation of GCD attack (Coppersmith et al., 1996), recovering RSA message encrypted with large exponent $e = 2^{16} + 1$.

In (Boneh et al., 2000), Boneh et al. proposed attacking n-bit RSA message, m, using meet-in-the-middle (MITM) attack. MITM attack is applied in two steps. A pre-computation step where the message is represented as $m = m_1 m_2$ with $m_1 \leq 2^{n_1}$ and

$m_2 \leq 2^{n_2}$. Hence, $c/m_2^e = m_1^e \ mod\ N$. A table of size 2^{n_1} has to be built containing the values $m_1^e \ mod\ N$ for all $m_1 \in 0,1,\cdots,2^{n_1} - 1$. Then, in the search step, we check for each $m_2 \in 0,1,\cdots,2^{n_2} - 1$, whether $c/m_2^e \ mod\ N$ is present in the table. Any collision reveals the message m. We implemented MITM attack (Boneh, Joux, and Nguyen 2000) using NTL ("NTL: A Library for Doing Number Theory" n.d.) library (Intel i5-8250U CPU 1.60 GHz, 8GB RAM), and we managed to recover a 40-bit message from Example A.1, Appendix A, encrypted with $e = 2^{16} + 1$ in 2.25 seconds for pre-computation step and 0.202 second for the searching step. Thus, from the analysis conducted we see that known lattice-based attacks against RSA private key, and against RSA messages practically use small public encryption exponent, a large part of the message to be known in advance, or a message to be broadcast. On the other hand, a non-lattice-based attack in (Coppersmith et al., 1996) has the cost of $O(e^2)$ for computing GCD (Micciancio 2016), where e is the RSA encryption exponent, while MITM (Boneh, Joux, and Nguyen 2000) has the cost of $O\left(n\sqrt{2^n}\right)$, where *n* is the message length in bits. The analysis of the attacks on RSA conducted above shows that they are not applicable for key-size greater than 829 bits, with p, q such that $p - 1, q - 1$ have large prime factors, encryption and decryption keys are greater than $N^{0.5}$, and $round(N^{0.5} - \lfloor p^{0.5} \rfloor \cdot \lfloor q^{0.5} \rfloor)$ is large. In Chapter 3 a new ciphertext-only attack on RSA is proposed.

2.3 Review of Class 3 (NTRU PKC)

In this section, NTRU PKC and its variants are reviewed.

2.3.1 Review of NTRU PKC

NTRU is described below according to (Jeffrey Hoffstein, Pipher, and Silverman 2014b, sec. 7.10.1).

NTRU has four positive integer parameters, (N, p, q, d), and uses the rings,

$$R = \frac{\mathbb{Z}[x]}{x^N - 1}, \qquad R_q = \frac{\mathbb{Z}_q[x]}{x^N - 1}, \qquad R_p = \frac{\mathbb{Z}_p[x]}{x^N - 1}.$$

A polynomial, $a(x) \in R$, loos as follows:

$$a(x) = \sum_{i=0}^{N-1} a_i x^i = [a_0, a_1, \dots, a_{N-1}] \tag{2.11}$$

NTRU uses trinary polynomials defined as follows,

$$\mathcal{T}(d_1, d_2) = \left\{ a(x) \in R: \begin{array}{c} a(x)\ has\ d_1\ coefficients\ equal\ to\ 1, \\ a(x)\ has\ d_2\ coefficients\ equal\ to\ \text{-}1, \\ a(x) has\ remained\ coefficients\ equal\ to\ 0 \end{array} \right\} \tag{2.12}$$

NTRU assumes that,

$$gcd(p, q) = gcd(N, q) = 1, q > (6d + 1)p \tag{2.13}$$

$p > 2$ is prime, $d \geq 1$ is an integer defining the structure used by NTRU polynomials, $f(x)$, $g(x)$, $r(x)$, are introduced below. NTRU uses two private keys: $f(x), g(x)$. The first private key, $f(x)$, is generated as follows:

$$f(x) = [f_0, f_1, \dots, f_{N-1}] \in \mathcal{T}(d + 1, d) \tag{2.14}$$

The private key (2.14), $f(x)$, must have inverses modulo p and q, that is, $F_p(x), F_q(x)$, respectively:

$$f \cdot F_q \equiv 1\ (mod\ q)\ and\ f \cdot F_p \equiv 1\ (mod\ p) \tag{2.15}$$

The private key, $g(x)$, is randomly chosen as follows:

$$g(x) \in \mathcal{T}(d, d) \tag{2.16}$$

The public key, $h(x)$, is computed using (2.15) and (2.16) as follows,

$$h(x) = F_q \cdot g\ mod\ q. \tag{2.17}$$

The plaintext message $m(x)$, is assumed to meet the following condition:

$$m(x) \in R_p. \tag{2.18}$$

Moreover, the coefficients of m are assumed to be center-lifted, i.e., to be in $\left(-\frac{1}{2}p, \frac{1}{2}p\right]$. A pseudo-randomly generated blinding polynomial, $r(x)$, is chosen as follows:

$$r(x) \in \mathcal{T}(d, d). \tag{2.19}$$

Ciphertext, $e(x)$, is computed as,

$$F_{h,p}^{NTRU}(m, r) = e(x) = p \cdot r \cdot h + m \in R_q\ . \tag{2.20}$$

Decryption in NTRU consists of Steps 1 and 2 described below.

Step 1: The first private key, $f(x)$, is applied to (2.20):

$$a = f \cdot e\ mod\ q = p \cdot r \cdot g + f \cdot m\ mod\ q, \tag{2.21}$$

Step 2: The second private key, F_p, is applied to (2.21) after a is center-lifted

$$m = a \cdot F_p\ mod\ p, \tag{2.22}$$

Decryption correctness condition is,

$$\forall i \in \{0, \dots, N-1\}, |a_i| < q/2 \tag{2.23}$$

Where a_i is the i-th coefficient of the polynomial $p \cdot r \cdot g + f \cdot m$.

In equation(2.22), F_p from (2.15) is used, and the contributor with factor p in (2.21) vanishes due to the constraints (2.13), (2.14)-(2.16), (2.19) imposed which guarantee that sum in the rightmost expression in (2.21) is a polynomial with coefficients strictly less than q, so that mod q operation, applied last in (2.21), does not change the coefficients.

Howgrave et. al., in (Howgrave-Graham, Silverman, and Whyte 2005; Howgrave-Graham, Silverman, et al. 2003) defined the following parameter settings for NTRU:

$p = 2$, $q > 4d + 1$ is prime, $N > 3k + 8$ is prime, k is the security parameter, and d is the minimal integer such that $\binom{N/2}{d/2}/\sqrt{N} > 2^k$, where $\binom{m}{n}$ is the number of combinations of n elements out of m. Secret polynomials, f, and g, are binary polynomials from D_f and D_g, with $d_f = d_g = d$ coefficients equal to one. Both f and g are invertible modulo q. The public key, h, is defined as in (2.17). A binary message, $m = D_m \in R_p$, is encrypted using a random binary polynomial r from D_r with the $d_r = \lfloor N/2 \rfloor$ ones according to (2.20). The decryption process is executed by (2.21), (2.22). The center-lifting to a, the LHS in (2.21) is done through the function $center()$. An implementation of $center()$, called $center1()$, provided in (Silverman et al., 2003, p. 4) follows:

1. Calculate $m(1)$ as $e(1) - p \cdot r(1) \cdot h(1) \; mod \; q$, reduced to the interval,

$$\frac{N-q}{2} \leq m(1) < \frac{N+q}{2}. \tag{2.24}$$

2. Denote a reduced to the interval $[0, q-1]$ by $\underline{a}$. The underline is intended to indicate the minimal possible interval.
3. Calculate $a(1)$. This will differ from $p \cdot r(1) \cdot h(1) + f(1) \cdot m(1)$ by $k \cdot q$, for some integer, k.
4. Add q to the lowest k entries of a to obtain a reduced into the correct interval.

NTRU decryption fails if the following condition does not hold,

$$Width(p \cdot g \cdot r + f \cdot m) < q. \tag{2.25}$$

where $Width(p(x)) = \max_{i=0,\ldots,N-1}(p_i) - \min_{i=0,\ldots,N-1}(p_i)$. Application of the $center()$ function to the left-hand side (LHS) makes the second equality in (2.21) true under condition (2.25). For the conditions imposed on NTRU parameters described

above, in particular, $p = 2$, $q > 4d + 1$, the second equality in (2.24)holds, and there is no need for centering.

2.3.2 Homomorphism of NTRU with Respect to Addition and Multiplication

NTRU is homomorphic w.r.t. two operations; addition and multiplication. From (2.20), let $e^{(i)}(x)$ be ta ciphertext encrypting the message $m^{(i)}$, $i = \{1, \dots, n\}$:

$$e^{(i)}(x) = p \cdot r^{(i)} \cdot h + m^{(i)} \in R_q \,. \tag{2.26}$$

Homomorphic addition and multiplication follow.

2.3.2.1 Homomorphic Addition

Let $e(x)$ be the ciphertext that encrypts the summation of $e^{(1)}$ and $e^{(2)}$ as follows:

$$e(x) = e^{(1)}(x) + e^{(2)}(x) = p \cdot h\left(r^{(1)} + r^{(2)}\right) + m^{(1)} + m^{(2)}.$$

It follows that

$$e(x) = \sum_{i=1}^{n} e^{(i)}(x) = p \cdot h \sum_{i=1}^{n} r^{(i)} + \sum_{i=1}^{n} m^{(i)} = p \cdot h \cdot r + m$$

Decryption of $e(x)$ resulting in $m = \sum_{i=1}^{n} m^{(i)}$ is performed by (2.21),(2.22). Decryption correctness condition is,

$$\forall i = \{0, \dots, N-1\}: |b_i| < q, \tag{2.27}$$

where b_i is the i-th coefficient of the polynomial $p \cdot g \sum_{i=1}^{n} r^{(i)} + f \sum_{i=1}^{n} m^{(i)}$, according to (2.23). Since r has small coefficients (see (2.19)), many homomorphic additions can be executed correctly as the growth of the noise, $\sum_{i=1}^{n} r^{(i)} = n\hat{r}$, is linear on n.

2.3.2.2 Homomorphic Multiplication

Multiplication of $e^{(1)}(x)$, $e^{(2)}(x)$ is performed as follows:

$$e(x) = e^{(1)}(x) \cdot e^{(2)}(x) = \left(p \cdot r^{(1)} \cdot h + m^{(1)}\right)\left(p \cdot r^{(2)} \cdot h + m^{(2)}\right)$$
$$= p \cdot h\left(p \cdot r^{(1)} \cdot r^{(2)} \cdot h + r^{(1)} \cdot m^{(2)} + r^{(2)} \cdot m^{(1)}\right) + m^{(1)}m^{(2)}$$
$$= p \cdot h \cdot r_x + m^{(1)}m^{(2)}$$

The decryption of $e(x)$ performed as follows:

Step1:

$$a = f^2 \cdot e(x) \; mod \; q = \left(p \cdot r^{(1)} \cdot g + f \cdot m^{(1)}\right)\left(p \cdot r^{(2)} \cdot g + f \cdot m^{(2)}\right) mod \; q$$

Step2: after center-lifting coefficients of polynomial a from Step1,

$$F_p^2 \cdot a \; mod \; p = m^{(1)} \cdot m^{(2)}.$$

Decryption correctness condition is:

$$\forall i = \{0, \dots, N-1\}: |c_i| < q, \qquad (2.28)$$

where c_i is the i-th coefficient of the polynomial $p \cdot h \cdot r_x + m^{(1)}m^{(2)}$, and $r_x = p \cdot r^{(1)} \cdot r^{(2)} \cdot h + r^{(1)} \cdot m^{(2)} + r^{(2)} \cdot m^{(1)} = \mathcal{O}(r^2)$. The growth of r_x is exponential in the number of multiplications which limits the number of allowed multiplication operations.

Thus, we conclude that NTRU is homomorphic w.r.t to addition and multiplication operations. NCM is needed when the thresholds defined in (2.27),(2.28) are exceeded.

NTRU works with degree N polynomials. The main problem NTRU faces is that it is susceptible to the lattice basis reduction attack (LBRA) using LLL algorithm. The LBRA using LLL algorithm solves the shortest vector problem (SVP) with exponential in N running time revealing the secret key because the private keys are selected as polynomials with small coefficients for the decryption correctness (Hoffstein et al., 1999). To overcome the problem of susceptibility, NTRU uses large N resulting in

high computational complexity (Hoffstein et al., 1998; "IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009).

NTRU variants (Bagheri et al., 2018; Banks & Shparlinski, 2002; Coglianese & Goi, 2005; Gaborit et al., 2002; Gaithuru & Salleh, 2017; Hoffstein et al., 2014b, p. 373; Howgrave-Graham et al., 2005; Howgrave-Graham, Silverman, Singer, et al., 2003; Jarvis & Nevins, 2015; Karbasi & Atani, 2015; Malekian et al., 2011; Malekian & Zakerolhosseini, 2010; M.G. & R., 2016; Seck & Sow, 2019; Stehlé & Steinfeld, 2011; Thakur & P., 2016; Thakur & Tripathi, 2017; Vats, 2009; B. Wang et al., 2018; Yassein & Al-Saidi, 2016; Yu et al., 2017), shown in Section 2.3.4, try minimizing NTRU computational complexity by extending the coefficients of the polynomials used or using matrices of polynomials that allow preserving the security level while decreasing the polynomial degree. The extreme case is a polynomial of zero degrees, that is integers modulo $q >> 1$, as used in the congruential public-key cryptosystem (CPKC) (Section 2.3.3.12.3.3), but CPKC with the NTRU encryption/encryption mechanism is insecure against LBRA by GLR (crackable in about 10 iterations) (Hoffstein et al., 2014b, pp. 373–376, 451). Therefore, the CPKC is considered as a toy model of NTRU because *"it provides the lowest dimensional introduction to the NTRU public-key cryptosystem"* (Hoffstein et al., 2014b, p. 374). The insecurity of CPKC stems from the choice of the private keys used as small numbers to provide decryption correctness. In Section 2.3.3, CPKC is reviewed, and LBRA using GLR is presented. Section 2.3.4 reviews known NTRU variants, and Section 2.3.5 reviews one of the main recent lattice attacks against NTRU, $IN - Lattice$ attack(Z. Yang et al., 2018b).

2.3.3 Review of CPKC and Lattice-Based Attack on CPKC

In this section, a review of the CPKC scheme (Hoffstein et al., 2014a) is provided, then an introduction of lattice-based attack on CPKC is given.

2.3.3.1 Review of CPKC Scheme

Two secret integers, f, and g, are defined as follows:

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2} \tag{2.29}$$

$$\gcd(f, q \cdot g) = 1 \tag{2.30}$$

where q is a public integer. The first secret value, f, has inverse modulo g and q, that is F_g and F_q, respectively, by virtue of (2.30),

$$1 = f \cdot F_g \; mod \; g, \quad 1 = f \cdot F_q \; mod \; q. \tag{2.31}$$

A public value, h, is computed using (2.29) and (2.31) as follows:

$$h = F_q \cdot g \; mod \; q. \tag{2.32}$$

Thus, CPKC has the private (secret) key, $SK = (f, g, F_g, F_q)$, and the public key, $PK = (h, q)$. The plaintext message, m, meets the following condition,

$$0 < m < \sqrt{q/4}. \tag{2.33}$$

A random integer, r, is chosen as follows:

$$0 < r < \sqrt{q/2}. \tag{2.34}$$

The ciphertext, e, is computed using (2.32)-(2.34) as follows:

$$e = F_h(m, r) = r \cdot h + m \; mod \; q \tag{2.35}$$

Decryption is described by Step 1 and 2 below:

Step1: Multiply the ciphertext (2.35) by f, getting:

$$a = f \cdot e \; mod \; q$$

$$= r \cdot f \cdot F_q \cdot g + f \cdot m \; mod \; q \tag{2.36}$$

Note that $a = r \cdot g + f \cdot m$ if (the remainder is allowed to be negative):

$$|r \cdot g + f \cdot m| < q \tag{2.37}$$

where (2.31), (2.32), and (2.35) are used. The CPKC decryption correctness condition (2.37) holds under conditions (2.29), (2.33) and (2.34):

$$|r \cdot g + f \cdot m| < \sqrt{q/2}\sqrt{q/2} + \sqrt{q/2}\sqrt{q/4} < q. \tag{2.38}$$

Thus, the parameters, f , g, and r, are selected as small compared to q (see (2.29), (2.33) and (2.30)) to meet the CPKC correctness decryption condition (2.37) used in Step 2 of the decryption. Step 2: Multiply (2.36) by F_g, getting:

$$m = a \cdot F_g \; mod \; g, \tag{2.39}$$

where (2.31) is used and the contributor with the factor g in (2.36) vanishes due to (2.37). Numerical Example C.1 of CPKC encryption/decryption is in Appendix C.

In Section 2.3.3.2, lattice basis reduction attack (LBRA) using GLR algorithm against the CPKC private key/message is explained.

2.3.3.2 Two-Dimensional CPKC Lattice

In (Hoffstein et al., 2014b, p. 376), it is shown that the CPKC private key recovery problem can be formulated as the shortest vector problem (SVP) in the two-dimensional lattice, $L(V_1, V_2)$. From (2.32), it can be noticed that for any pair of integers, F and G, satisfying:

$$G = F \cdot h \; mod \; q, \;\; F = \mathcal{O}\left(\sqrt{q}\right), \;\; G = \mathcal{O}\left(\sqrt{q}\right) \tag{2.40}$$

(F, G) is likely to serve as the first two components, f, and g, of the private key, SK (Hoffstein et al., 2014b, p. 376). Equation (2.40) can be written as $F \cdot h \; + \; q \cdot n = G$, where n is an integer. Therefore, our task is to find a pair of comparatively small by absolute value integers, (F, G), such that:

$$F \cdot V_1 + n \cdot V_2 = (F, G), \tag{2.41}$$

where $V_1 = (1, h)$ and $V_2 = (0, q)$ are basis vectors, at least one of them having the Euclidean norm of order $\mathcal{O}(q)$. Similarly, the CPKC message recovery problem can be formulated as the SVP in the two-dimensional lattice, $L(V_1, V_2)$, where V_1 and V_2 are from (2.41). It can be also noticed from (2.35) that for any pair of integers, (RR, EM), satisfying:

$$EM = RR \cdot h \bmod q, RR = \mathcal{O}(\sqrt{q}), EM = \mathcal{O}(\sqrt{q}), \tag{2.42}$$

(RR, EM) is likely to serve as the vector $(r, e - m)$ since the encryption Equation (2.35) can be written as $r \cdot h + q \cdot n = e - m$, where n is an integer. Therefore, our task is to find a pair of comparatively small by absolute value integers, (RR, RM), such that:

$$RR \cdot V_1 + n \cdot V_2 = (RR, EM). \tag{2.43}$$

We aim to find the shortest vector w from $L(V_1, V_2)$ using LLL that might disclose $(r, e - m)$ if e and r are of the order of $\mathcal{O}(\sqrt{q})$. Comparing (2.41) and (2.43), it is noticed that they are the same up to the unknowns' names used, and hence, finding the shortest vector in $L(V_1, V_2)$ may reveal either the private key components, $(F, G) = (f, g)$, or the message related vector, $(RR, EM) = (r, e - m)$.

2.3.3.3 Gaussian Lattice Reduction Attack on CPKC Key/Message

Maple code (shown in Code B.1, Appendix B) of LBRA by GLR on CPKC private key/message returned as the shortest vector $w = v_1$ of the lattice $L(V_1, V_2)$, where V_1, V_2 are from (2.41). Example C.2 in Appendix C, provides an example of a GLR attack against CPKC. LBRA by GLR finds in 9 iterations the shortest vector, $v_1 = (f, g)$ corresponds to the private key components, (f, g), because they were selected small, having order $\mathcal{O}(\sqrt{q})$ values according to (2.40).

2.3.4 Review of Known NTRU Variants

Many variants of NTRU have been proposed and studied recently, targeting further decreasing its computational complexity. All these variants work with polynomials and mainly differ in the choice of their coefficients, the ring defining polynomial, or the polynomials used as the entries of such structures as matrices. The NTRU variants overview follows.

2.3.4.1 NTRU Variants Differing in the Choice of Their Coefficients

In (Jarvis & Nevins, 2015), the NTRU variant, ETRU, was proposed working with polynomials over Eisenstein integer coefficients and was faster than NTRU in encryption/decryption by 1.45/1.72 times. Karbasi and Atani (Karbasi & Atani, 2015) modified ETRU, called ILTRU (Karbasi and Atani 2015) so that it works with irreducible cyclotomic polynomial over Eisenstein integer coefficients. An NTRU variant, BITRU, working with polynomials over so-called binary numbers, usually known as complex numbers, was proposed in (M.G. & R., 2016). An NTRU variant, QTRU, working with polynomials over hyper-complex four-component numbers, quaternions, was proposed in (Malekian et al., 2011). Furthermore, Bagheri and colleagues proposed an NTRU variant, BQTRU, working over quaternions, but with bivariate polynomials, seven times faster than NTRU in encryption (Malekian et al., 2011). A variant of NTRU working with polynomials over eight-component hyper-complex numbers, octonions(Malekian & Zakerolhosseini, 2010)alekian & Zakerolhosseini, 2010). In (Yassein & Al-Saidi, 2016), NTRU variant, HXDTRU, was proposed working with polynomials over 16-component hyper-complex numbers, hexadecnions, also known as sedenions (*Sedenion - Wikipedia*, n.d.). Furthermore, a variant of NTRU working with polynomials over 16-component hyper-complex numbers, sedenions, was proposed in (Thakur & Tripathi, 2017). A variant of NTRU,

called CTRU, working with polynomials, the coefficients of which are also polynomials in one variable over a binary field, was proposed in (Gaborit et al., 2002). Furthermore, a variant of NTRU working with polynomials, the coefficients of which are polynomials in one variable over a rational field, called BTRU, was proposed in (Thakur & P., 2016).

2.3.4.2 NTRU Variants Working with Different Rings

An NTRU variant that works with polynomials with prime cyclotomic rings was proposed (Yu et al., 2017). A variant of NTRU working with non-invertib(Banks & Shparlinski, 2002) in (Banks & Shparlinski, 2002).

2.3.4.3 NTRU Variants Working with Polynomials Inside More Complicated Structures

An NTRU variant working with square matrices of polynomials was proposed in (Coglianese & Goi, 2005) and was shown to be 2.5 times better than NTRU encryption and decryption time. An NTRU variant, called NNRU, working with polynomials also being entries of square matrices forming a non-commutative ring, was proposed in (Vats, 2009). Apart from the polynomial variants, an NTRU-like cipher over the ring of integers, called ITRU, was proposed in (Gaithuru & Salleh, 2017) without referencing CPKC (Hoffstein et al., 2014b). In ITRU (Gaithuru & Salleh, 2017, pp. 34, Table 1), the NTRU model specified above was given, but a model for the proposed ITRU was not defined. Its Algorithm 1, (Gaithuru & Salleh, 2017, p. 35), describes the key generation, and hence, it shall be made by the key owner (receiver). On the other hand, in (Gaithuru & Salleh, 2017, p. 37), the most important parameter, q, was selected by the sender (which encrypts a message using the public key, $h' = 423{,}642$, and a random value, $r' = 19$, in (Gaithuru & Salleh, 2017, p. 37, eq. (19)) with the help of the private keys, f', g', which contradicts the NTRU model: the secret key is

known to only the key owner that uses the private key only for decryption, whereas the public key is used for encryption by the public user only.

2.3.5 Review of $IN-Lattice$ Attack on NTRU Private Keys

In this section, lower dimension lattice (than the normal $2N$-dimensional NTRU lattice $\mathcal{L}^{ntru}$ (Hoffstein et al., 1998, p. 273, 2014b, p. 425)) attack on NTRU (Z. Yang et al., 2018b), hereafter ($IN-Lattice$ Attack), is briefly reviewed. In (Z. Yang et al., 2018b), a class of lower dimension lattices (lower than $2N$) is constructed, called $IN-Lattice$, and a new lattice attack on NTRU is presented using these lattices.

Definition 2.6: (Hoffstein et al., 2014b, p. 402)**.** Let $\mathcal{L}$ be a lattice of dimension n. The Gaussian expected shortest length is:

$$\sigma(\mathcal{L}) = \sqrt{\frac{n}{2\pi e}}(\det(\mathcal{L}))^{1/n}. \tag{2.44}$$

More precisely, if $\epsilon > 0$ is fixed, then for all sufficiently large n, a randomly chosen lattice of dimension n will satisfy:

$$(1-\epsilon)\sigma(\mathcal{L}) \leq \|v_{shortest}\| \leq (1+\epsilon)\sigma(\mathcal{L}) \tag{2.45}$$

where $v_{shortest}$ is the shortest nonzero vector in $\mathcal{L}$. For $2N$-dimension lattice,

$$\|v_{shortest}\| \leq \frac{(1+\epsilon)(\det(\mathcal{L}))^{1/(2N)}}{\sqrt{2\pi e}}\sqrt{2N} < \tag{2.46}$$

Let $\tau > (1+\epsilon)(\det(\mathcal{L}))^{1/(2N)}/\sqrt{2\pi e}$. Then,

$$\|v_{shortest}\| < \tau\sqrt{2N} \tag{2.47}$$

Definition 2.7: (Z. Yang et al., 2018b)**.** The NTRU problem is defined by four parameters: a ring R of rank N and endowed with an inner product, a modulus q, a uniform distribution D in set $\ell(d_1, d_2)$ which contains all polynomials with d_1 coefficients equal to 1, d_2 coefficients equal to -1, and the rest 0., and a target norm τ

in (2.47). Precisely, NTRU(R, q, D, τ) is the problem of, given $h = [gf^{-1}]_q$ (conditioned on f being invertible mod q) for $f, g \leftarrow D$, finding a vector $(x, y) \in R^2$ such that $(x, y) \neq (0, 0)\ mod\ q$ and of Euclidean norm less than $\tau\sqrt{2N}$ in the lattice

$$\mathcal{L}^{ntru} = \{(x, y)\ such\ that\ hx - y = 0\ mod\ q\}. \tag{2.48}$$

In other words, the problem of NTRU finds the private key (f, g) as the shortest vector (x, y) in the lattice $\mathcal{L}^{ntru}$.

Let $f \cdot h$ multiplied in $R = \mathbb{Z}[x]/(x^N - 1)$, then can be represented by

$$(f_0, f_1, \dots, f_{N-1}) \begin{bmatrix} h_0 & h_1 & \cdots & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_0 \end{bmatrix} = fH$$

The circular matrix of h can be written as follows:

$$H = (h_0^T, h_1^T, \dots, h_{N-1}^T). \tag{2.49}$$

where h_i^T $(0 \leq i \leq N-1)$ is the i-th column vector in H. Let $v = (v_0, v_1, \dots, v_{N-1}) \in \mathbb{Z}^N$. Then we define $v^{ls(l)} = (v_l, v_{l+1}, \dots, v_{l-1})$ as cycle left-shift of $\boldsymbol{v}$ by l positions.

Definition 2.8: (Z. Yang et al., 2018b) Let I be a subset of $\{1,2, \dots, N\}$ such that g_{i+k} is 0 for all $i \in I$, where k is a constant integer belonging to $\{1,2, \dots, N\}$. And le $t = \#I$. Then, an *IN-Lattice* $\mathcal{L}_I$ with size t is defined by

$$\mathcal{L}_I = \{x \in \mathbb{Z}^N : \forall_i \in I, \quad x \cdot h_i \equiv 0\ mod\ q\}.$$

The new $IN - Lattice$ attack is introduced in (Z. Yang et al., 2018b)

Algorithm 2.1: $IN - Lattice$ Attack (Z. Yang et al., 2018b, p. 2)

Require: Fix $N, q, d_g h$, and find the probability $\Pr(\boldsymbol{f}^{ls(k)} \in \mathcal{L}_I)$ according to (2.50);

Ensure: A valid private key $\boldsymbol{f}'$;

1.$t \leftarrow 2$;

2.**While** $t < N$ **do**

3. count ← 1;

4. **While** count <=$\lceil 1/\Pr(\boldsymbol{f}^{ls(k)} \in \mathcal{L}_I) \rceil$ **do**

5. Randomly choose a subset I of $\{1,2,\ldots,N\}$ sch that $\#I = t$;

6. Construct an $IN - Lattice\ \mathcal{L}_I$ with size t;

7. Reduce $\mathcal{L}_I$;

8. **if** the reduced basis contains a vector $\boldsymbol{v}$ which can be used to decrypt **then**

9. $\boldsymbol{f}' = \boldsymbol{v}$;

10: Output $\boldsymbol{f}'$, t and break;

11: **end if**

12: count=count+1;

13: **end while**

14: $t \leftarrow t + 1$;

15:**end while**

The probability of having the $\boldsymbol{f}^{ls(k)}$ belongs to $\mathcal{L}_I$ is calculated according to (2.50),

$$\Pr(f^{ls(k)} \in \mathcal{L}_I) = 1 - \left(1 - \prod_{i=0}^{2d_g-1}\left(1 - \frac{t}{N-i}\right)\right)^N, \qquad (2.50)$$

where $2d_g$ is the Hamming weight of g and $t = \#I$.

The efficiency of $IN - Lattice$ attack is investigated in Section 4.2, and a new attack is proposed in Section 4.1.

2.4 Review of Class 4 (HE1N)

HE1N (Dyer et al., 2019, p. 8), the most recent HE scheme in this classis reviewed.

2.4.1 Review of HE1N

HE1N is defined as follows:

2.4.1.1 Parameter Settings

Let $m_1, m_2, \dots, m_n$ be distributed in $[0, M)$ according to a probability distribution D with entropy ρ, where ρ is not large enough to negate a brute force attack. Therefore, the entropy of the plaintext is increased by adding "noise" term to the ciphertext. This will be a multiple s (from 0 to k) of an integer k, chosen so that the entropy $\rho' = \rho + \log k$ is large enough to negate a brute force guessing attack. Let λ be a security parameter, measured in bits. Choose primes p, q such that $p \in \left[2^{\lambda-1}, 2^{\lambda}\right]$, and $q \in [2^{\eta-1}, 2^{\eta}]$, where $\eta \approx \lambda^2/\rho' - \lambda$, and $p > (n+1)^d (M + k^2)^d$, where d is the degree of the polynomial P homomorphically computed over ciphertexts, $(m_1 + s_1 k, m_2 + s_2 k, \dots, m_N + s_n k)$, such that $P(m_1 + s_1 k, m_2 + s_2 k, \dots, m_N + s_n k) < p$, when $s_1, s_2, \dots, s_n \in [0, k)$. Parameter k is randomly chosen such that $k > (n+1)^d M^d$. Key generation algorithm and parameter generation algorithm is presented in Algorithm 2.2 and Algorithm 2.3 respectively.

Algorithm 2.2: Key generation algorithm

Input: $\lambda \in S, \rho \in \mathbb{Z}$: entropy of inputs, $\rho' \in \mathbb{Z}$: effective entropy of inputs,

Output: (k, p): secret key

1. $p \leftarrow \left[2^{\lambda-1}, 2^{\lambda}\right]$;
2. $v \leftarrow \rho' - \rho$
3. $k \leftarrow [2^{v-1}, 2^{v}]$
4. **Return** (k, p)

Algorithm 2.3: Parameter generation algorithm

Input: $\lambda \in S, \rho' \in \mathbb{Z}$: effective entropy of inputs, (k, p): secret key

Output: modulus $\in \mathbb{Z}$: modulus for encryption and homomorphic operations

1. $\eta \leftarrow \lambda^2/\rho' - \lambda$;
2. $q \leftarrow [2^{\eta-1}, 2^{\eta}]$
3. modulus $\leftarrow pq$
4. **Return** modulus

where S is the security parameter space.

2.4.1.2 Encryption

The plaintext m is encrypted using Algorithm 2.4

Algorithm 2.4: The encryption algorithm

Input: $m \in M$, (k, p): a secret key, modulus: public modulus

Output: $c \in C$

1. $q \leftarrow \text{modulus}/p$;
2. $r \leftarrow [1, q)$
3. $s \leftarrow [0, k)$
4. $c \leftarrow m + sk + rp \; mod$ modulus
5: **Return** c

The decryption process of ciphertext c follows in Subsection 2.4.1.3

2.4.1.3 Decryption

Ciphertext c is decrypted using Algorithm 2.5

Algorithm 2.5: Decryption algorithm

Input: $c \in C$, p:secret key

Output: $m \in \mathcal{M}$

1. $m \leftarrow (c \bmod p) \bmod k$;
2. **Return** m

Decryption works since by definition $m + sk < p$ and $m < k$. Appendix D presents an example of the encryption-decryption process of HE1N scheme. Chapter 5, considers homomorphism of HE1N, the condition for HE1N to be homomorphic for any polynomial P of degree d is set, and several attacks are proposed against HE1N.

2.4.2 Homomorphism of HE1N With Respect to Addition and Multiplication

From Algorithm 2.4, let $c^{(i)} = m^{(i)} + s^{(i)}k + r^{(i)}p \bmod pq, i = \{1, \dots, n\}$. Homomorphic addition and multiplication operations follow

2.4.2.1 Homomorphic Addition

Let c be the ciphertext encrypts the summation of $c^{(1)}$ and $c^{(2)}$ as follows:

$$c = c^{(1)} + c^{(2)} = m^{(1)} + m^{(2)} + \left(s^{(1)} + s^{(2)}\right)k + \left(r^{(1)} + r^{(2)}\right)p \bmod pq$$

It follows that

$$c = \sum_{i=1}^{n} c^{(i)} = \sum_{i=1}^{n} m^{(i)} + k\sum_{i=1}^{n} s^{(i)} + p\sum_{i=1}^{n} r^{(i)} \bmod pq$$

The decryption of c is performed by Algorithm 2.5. Decryption works in the condition that:

$$\sum_{i=1}^{n} m^{(i)} + k\sum_{i=1}^{n} s^{(i)} < p. \tag{2.51}$$

For small $s^{(i)}, i = \{1, \dots, n\}$, many homomorphic additions can be executed correctly as the growth of the noise, $\sum_{i=1}^{n} s^{(i)} = n\hat{s}$, is linear on s.

2.4.2.2 Homomorphic Multiplication

Multiplication of $c^{(1)}$, $c^{(2)}(x)$ is performed as follows

$$c = c^{(1)} \cdot c^{(2)} = \left(m^{(1)} + s^{(1)}k + r^{(1)}p\right)\left(m^{(2)} + s^{(2)}k + r^{(2)}p\right)$$

$$= m^{(1)}m^{(2)} + m^{(1)}s^{(2)}k + m^{(1)}r^{(2)}p + s^{(1)}km^{(2)} + s^{(1)}ks^{(2)}k + s^{(1)}kr^{(2)}p$$
$$+ r^{(1)}pm^{(2)} + r^{(1)}ps^{(2)}k + r^{(1)}p\, r^{(2)}p =$$
$$m^{(1)}m^{(2)} + \left(m^{(1)}s^{(2)} + s^{(1)}m^{(2)} + s^{(1)}ks^{(2)}\right)k$$
$$+ \left(m^{(1)}r^{(2)} + s^{(1)}kr^{(2)} + r^{(1)}m^{(2)} + r^{(1)}s^{(2)}k + r^{(1)}\, r^{(2)}p\right)p$$
$$= m^{(1)}m^{(2)} + s_x k + r_x p.$$

The decryption of c performed by Algorithm 2.5. Decryption correctness condition is,

$$s_x < p, \tag{2.52}$$

where $s_x = m^{(1)}s^{(2)} + s^{(1)}m^{(2)} + s^{(1)}s^{(2)}k = \mathcal{O}(r^2)$. The growth of s_x is exponential in the number of multiplications which limits the number of allowed multiplication operations.

Thus, we conclude that HE1N is homomorphic w.r.t to addition and multiplication operations. NCM is needed when the thresholds defined in (2.51),(2.52) are exceeded.

2.5 Review of Class 6 (RLWE-NCM-CSCM PKC)

In this section, a HE scheme based on ring learning with errors (R-LWE) by Brakerski et. al. (Brakerski & Vaikuntanathan, 2011b), is considered. The scheme as shown in columns 4, 5 of Table 2.1 needs NCM and CSCM, therefore, hereafter we called it RLWE-NCM-CSCM. RLWE-NCM-CSCM (Brakerski & Vaikuntanathan, 2011b) has two versions; symmetric and asymmetric. In this section, the symmetric version is analyzed for simplicity.

The plaintext $m \in \mathbb{Z}_t[x]/(x^n + 1)$ is encrypted by,

$$c = \binom{c_0}{c_1} = \binom{a \cdot s + t \cdot e + m}{-a} \in R_q^2, \tag{2.53}$$

where $a \leftarrow R_q = \mathbb{Z}_q[x]/(x^n + 1)$, n is a power of 2, q is a prime number, and a prime $t \in \mathbb{Z}_q^+$, e and s are polynomials which coefficients are sampled from a discrete Gaussian distribution χ. The ciphertext (2.53) has two components. The decryption process is performed in two steps as follows:

Step1: apply $< c, sk >$ operation, where $<>$ is the inner product, c is the ciphertext (2.53), $sk = \binom{1}{s}$.

$$c_0 + c_1 \cdot s \; mod \; (x^n + 1) \; mod \; q \; = t \cdot e + m \; mod \; t \tag{2.54}$$

Step2: apply modulo t operation to the output from Step1,

$$m = t \cdot e + m \; mod \; t \tag{2.55}$$

Decryption correctness condition is,

$$\forall i \in \mathbb{Z}_n : |t \cdot e_i + m_i| \leq q/2, \tag{2.56}$$

where e_i, m_i are i-th coefficients of e and m respectively.

Example E.1 and Example E.2, Appendix E, are examples of failing decryption when condition (2.56) is not satisfied, and an example of successful decryption when condition (2.56) is satisfied, respectively.

Let $c^{(i)}$ be the ciphertext encrypting message $m^{(i)}$:

$$c^{(i)} = \begin{pmatrix} c_0^{(i)} \\ c_1^{(i)} \end{pmatrix} = \begin{pmatrix} a^{(i)} \cdot s + t \cdot e^{(i)} + m^{(i)} \\ -a^{(i)} \end{pmatrix}, i = 1, .., k \tag{2.57}$$

Homomorphic addition and multiplication of $c^{(1)}$, $c^{(2)}$ follow.

2.5.1 Homomorphic Addition

Homomorphic addition of $c^{(1)}$, $c^{(2)}$ is performed as follows:

$$C_{add} = \begin{pmatrix} C_{add,0} \\ C_{add,1} \end{pmatrix} = \begin{pmatrix} c_0^{(1)} + c_0^{(2)} \\ c_1^{(1)} + c_1^{(2)} \end{pmatrix} =$$

$$\begin{pmatrix} \left(a^{(1)} + a^{(2)}\right)s + t\left(e^{(1)} + e^{(2)}\right) + m^{(1)} + m^{(2)} \\ -\left(a^{(1)} + a^{(2)}\right) \end{pmatrix} \in R_q^2.$$

It follows that,

$$c_{add} = \sum_{i=1}^{k} c^{(i)} = \begin{pmatrix} \sum_{i=1}^{k} \left(sa^{(i)} + te^{(i)} + m^{(i)}\right) \\ -\sum_{i=1}^{k} a^{(i)} \end{pmatrix} = \begin{pmatrix} s \cdot \bar{a} + t \cdot \bar{e} + m \\ -\bar{a} \end{pmatrix}, \quad (2.58)$$

Decryption of c_{add} resulting in $m = \sum_{i=1}^{n} m^{(i)}$ is performed by (2.54). The decryption correctness condition is $\forall i \in \mathbb{Z}_n: |t \cdot \bar{e}_i + m_i| \leq q/2$ according to (2.54), where $\bar{e}_i, m_i$ are i-th coeffiecients of $\bar{e}$ and m respectively. For e such that $\forall i \in \mathbb{Z}_n: e_i \ll q$, many homomorphic additions can be executed correctly as the growth of the noise $t\sum_{i=1}^{k} e^{(i)} = tk\hat{e}$ is linear on k.

2.5.2 Homomorphic Multiplication

From (2.54), (2.57), $c_0^{(i)} + c_1^{(i)} \cdot s = t \cdot e^{(i)} + m^{(i)}$, for $i = 1, 2$. Then, it holds that

$$\left(t \cdot e^{(1)} + m^{(1)}\right)\left(t \cdot e^{(2)} + m^{(2)}\right) = \left(c_0^{(1)} + c_1^{(1)} \cdot s\right)\left(c_0^{(2)} + c_1^{(2)} \cdot s\right) \quad (2.59)$$

$$= c_0^{(1)} \cdot c_0^{(2)} + \left(c_0^{(1)} \cdot c_1^{(2)} + c_1^{(1)} \cdot c_0^{(2)}\right) \cdot s + c_1^{(1)} \cdot c_1^{(2)} \cdot s^2 \quad (2.60)$$

$$= c_{mult,0} + c_{mult,1} \cdot s + c_{mult,2} \cdot s^2 \quad (2.61)$$

From (2.59)-(2.61), we see that a ciphertext, c_{mult} is defined in (2.62) can be decrypted using $sk = \begin{pmatrix} 1 \\ s \\ s^2 \end{pmatrix}$,

$$C_{mult} = \begin{pmatrix} C_{mult,0} \\ C_{mult,1} \\ C_{mult,2} \end{pmatrix} = \begin{pmatrix} c_0^{(1)} \cdot c_0^{(2)} \\ c_0^{(1)} \cdot c_1^{(2)} + c_1^{(1)} \cdot c_0^{(2)} \\ c_1^{(1)} \cdot c_1^{(2)} \end{pmatrix} \in R_q^3. \quad (2.62)$$

Note that the size (the number of components) of the ciphertext (2.62) increased due to multiplication from two to three. Thus, decryption represented by formula (2.54) can't be used to decrypt the ciphertext (2.62), and new decryption must be defined to allow 3-dimension vectors as follows:

$$m = m^{(1)}m^{(2)} = C_{mult,0} + C_{mult,1} \cdot s + C_{mult,2} \cdot s^2 \; mod \; t. \qquad (2.63)$$

$$= t\left(te^{(1)}e^{(2)} + e^{(1)}m^{(2)} + e^{(2)}m^{(1)}\right) + m^{(1)}m^{(2)} \; mod \; t = t\bar{e} + m \; mod \; t.$$

Decryption correctness condition is,

$$\forall i \in \mathbb{Z}_n : |t \cdot \bar{e}_i + m_i| \leq q/2, \qquad (2.64)$$

where $\bar{e}_i$, m_i are the i-th coefficients of $\bar{e} = te^{(1)}e^{(2)} + e^{(1)}m^{(2)} + e^{(2)}m^{(1)} = O(\hat{e}^2)$, and $m = m^{(1)}m^{(2)}$. The growth of the noise $\bar{e}$ is exponential in the number of multiplications which limits the number of allowed multiplication operations. This restricts the depth d of the homomorphic functions evaluated on the ciphertexts. Brakerski and Vaikuntanathan bootstrapped their scheme using Gentry's method (Gentry, 2009b) to upgrade RLWE-NCM-CSCM into FHE. In (Brakerski & Vaikuntanathan, 2011a), Brakerski and Vaikuntanathan proposed a new method to control the increase in the ciphertext size, called "re-linearization" (Brakerski & Vaikuntanathan, 2011a), in which the secret key, s and s^2 are encrypted using a new secret key, called recryption key, u, as follows:

$$c^{(s)} = \begin{pmatrix} c_0^{(s)} \\ c_1^{(s)} \end{pmatrix} = \begin{pmatrix} a^{(3)}u + te^{(3)} + s \\ -a^{(3)} \end{pmatrix}, \text{and} \quad c^{(s^2)} = \begin{pmatrix} c_0^{(s^2)} \\ c_1^{(s^2)} \end{pmatrix}$$

$$= \begin{pmatrix} a^{(4)}u + te^{(4)} + s^2 \\ -a^{(4)} \end{pmatrix}, \qquad (2.65)$$

where $a^{(3)}, a^{(4)} \leftarrow R_q$ and $e^{(3)}, e^{(4)} \leftarrow \chi$ From (2.65) one obtains,

$$se = c_0^{(s)} + c_1^{(s)}u = te^{(3)} + s, \qquad (2.66)$$

and,

$$se2 = c_0^{(s^2)} + c_1^{(s^2)}u = te^{(4)} + s^2. \tag{2.67}$$

From (2.63), (2.66), (2.67):

$$C_{mult,0} + C_{mult,1} \cdot s + C_{mult,2} \cdot s^2 \; mod \; t;$$

$$= C_{mult,0} + C_{mult,1} \cdot se + C_{mult,2} \cdot se2 \; mod \; t, \tag{2.68}$$

by substituting (2.66), (2.67) in (2.68):

$$C_{mult,0} + C_{mult,1} \cdot se + C_{mult,2} \cdot se2$$

$$= C_{mult,0} + C_{mult,1} \cdot \left(c_0^{(s)} + c_1^{(s)}u\right) + C_{mult,2} \cdot \left(c_0^{(s^2)} + c_1^{(s^2)}u\right)$$

$$= C_{mult,0} + C_{mult,1} \cdot c_0^{(s)} + C_{mult,2} \cdot c_0^{(s^2)} + \left(C_{mult,1} \cdot c_1^{(s)} + C_{mult,2} \cdot c_1^{(s^2)}\right)u. \tag{2.69}$$

From (2.69), a two-component vector,

$$c = \begin{pmatrix} C_{mult,0} + C_{mult,1} \cdot c_0^{(s)} + C_{mult,2} \cdot c_0^{(s^2)} \\ C_{mult,1} \cdot c_1^{(s)} + C_{mult,2} \cdot c_1^{(s^2)} \end{pmatrix}, \tag{2.70}$$

can be decrypted using (2.54) for $sk = u$. Thus, re-linearization is represented by (2.70), and a chain of keys is needed for several multiplications. And to solve the issue of growing the noise, Brakerski and Vaikuntanathan (Brakerski & Vaikuntanathan, 2011a), proposed the Dimension-Modulus Reduction method instead of Gentry's Bootstrapping (Gentry, 2009b). In (Brakerski et al., 2014) Brakerski et. al. showed that modulus reduction and re-linearization (Brakerski & Vaikuntanathan, 2011a) have worse performance than Gentry's bootstrapping (Gentry, 2009b). Wang et al. (X. Wang et al., 2018) proposed an FHE based on L-FHE using modulus switching and key switching to overcome the noise growth problem. In (Gao, 2018), Gao presents an FHE scheme based on LFHE using bootstrapping. Thus, we see that all LWE-based FHE are LFHE upgraded to FHE using Gentry's bootstrapping or other tools such as

modulus switching. In Chapter 7, RLWE-CSCM is presented, the first FHE scheme that doesn't affect noise growth by structure. Therefore, it does not need a noise control mechanism. RLWE-CSCM still needs a ciphertext size control mechanism; therefore, we present two different mechanisms to control the growth of the ciphertext.

2.6 Review of Class 7 (Homomorphic Scheme Using Ideal Lattices)

This section reviews a homomorphic scheme using ideal lattices proposed in (Gentry, 2009b) from Class 7. The following description is adapted from (Gentry & Halevi, 2011).

Gentry's Ideal lattice scheme can be seen as a GGH-type scheme over ideal lattices (Gentry & Halevi, 2011). The ring $R = \frac{\mathbb{Z}[x]}{x^n+1}$ is a set of polynomials of degree $n-1$ with integer coefficients. A polynomial from R can be represented as a coefficient vector in $\mathbb{Z}^n$, e.g., the polynomial $p(x) = 3x + 1$ can be represented by the vector $\vec{p} = (1, 3)$, respectively, a vector $\vec{v} = (5, -2)$ can be equivalently represented by a polynomial $p(x) = -2x + 5$. That is why order $n-1$ polynomials and n-dimensional vectors will be used hereafter interchangeably.

The ideal $J = (\vec{v}) \subset R$, is a principal ideal set by choosing a vector $\vec{v} \in \mathbb{Z}^n$ at random. The basis matrix V of the ideal $J = (\vec{v})$ is the rotational matrix $V = \{\vec{v}_i = \vec{v} \cdot x^i \ mod\ (x^n + 1) : i \in [0, n-1]\}$.

The public key consists of a "bad" basis B_{pk} of an ideal lattice J. The public key, B_{pk} is the HNF of the matrix V. The secret key is the "good" basis $B_{sk} = V$ of the ideal lattice J.

To encrypt the bit b, two steps are used. In the first step, $\vec{a}$ is calculated by (2.71)

$$\vec{a} = 2\vec{u} + b\vec{e}_1 \in \mathbb{Z}^n, \tag{2.71}$$

Where $u_i \in \{-1,0,1\}$ is a random vector, $\vec{e}_1 = (1,0,\dots,0) \in \mathbb{Z}^n$ is the unit vector. Then, in the second step, the ciphertext is calculated by (2.72),

$$\vec{c} = \vec{a}\ mod\ B_{pk}. \tag{2.72}$$

When $\vec{a}$ has a small enough norm, i.e., less than $\lambda_1(L)/2$, it can be computed from the difference between $\vec{c}$ and the closest lattice vector. Thus,

$$\vec{a} = \vec{c}\ mod\ B_{sk}. \tag{2.73}$$

where B_{sk}, the secret key, is the good basis of J. The bit b can be recovered by applying modulo 2 to $\vec{a}(1)$, where $\vec{a}(1)$ is the first element of the vector $\vec{a}$.

$$b = \vec{a}(1)\ mod\ 2. \tag{2.74}$$

The reason for decryption works is that, if the parameters are chosen correctly, then the parallelepiped $\mathcal{P}(B_{sk})$ of the secret key will be a "plump" parallelepiped that contains a sphere of radius bigger than $\|\vec{a}\|$, so that $\vec{a}$ is the point inside $\mathcal{P}(B_{sk})$. On the other hand, the parallelepiped $\mathcal{P}(B_{pk})$ of the public key will be very skewed, and will not contain a sphere of large radius (Gentry & Halevi, 2011, p. 134). Therefore B_{pk} can't be used to retrieve $\vec{a}$.

Example 2.1 Example of Homomorphic Scheme Using Ideal Lattices Encryption/ Decryption

In this example, the homomorphic scheme (Gentry, 2009b) is considered using the settings in (Gentry & Halevi, 2011),

1- Parameter Settings:

The scheme uses the ring $R = \mathbb{Z}[x]/(x^n + 1)$, for $n = 2$. The ideal $J = (\vec{v})$ is generated by $\vec{v}$, where the vector $\vec{v}$ be selected randomly according to (2.75),

$$\vec{v} = (7,3) \in R. \tag{2.75}$$

Recall that the vector $\vec{v}$ can be represented as polynomial, $v(x) = 3x + 7$. Thus, the basis matrix V of the ideal $(\vec{v})$ in (2.76) is the rotational matrix $V = \{\vec{v}_i = \vec{v} \cdot x^i \ mod\ (x^n + 1) : i \in [0, n-1]\}$,

$$V = \begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}. \tag{2.76}$$

The public key, B_{pk} in (2.77), is the bad basis, HNF of (2.76) found by Maple,

$$B_{pk} = HNF(V) = \begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}. \tag{2.77}$$

$$V := Matrix([[7, 3], [-3, 7]])$$

$$\begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$$HermiteForm(V)$$

$$\begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}$$

We can see from Figure 2.3, that Basis $B_{sk} = V$ (left) is better than $B_{pk} = HNF(V)$(right). Hadamard ratio for both bases follows.

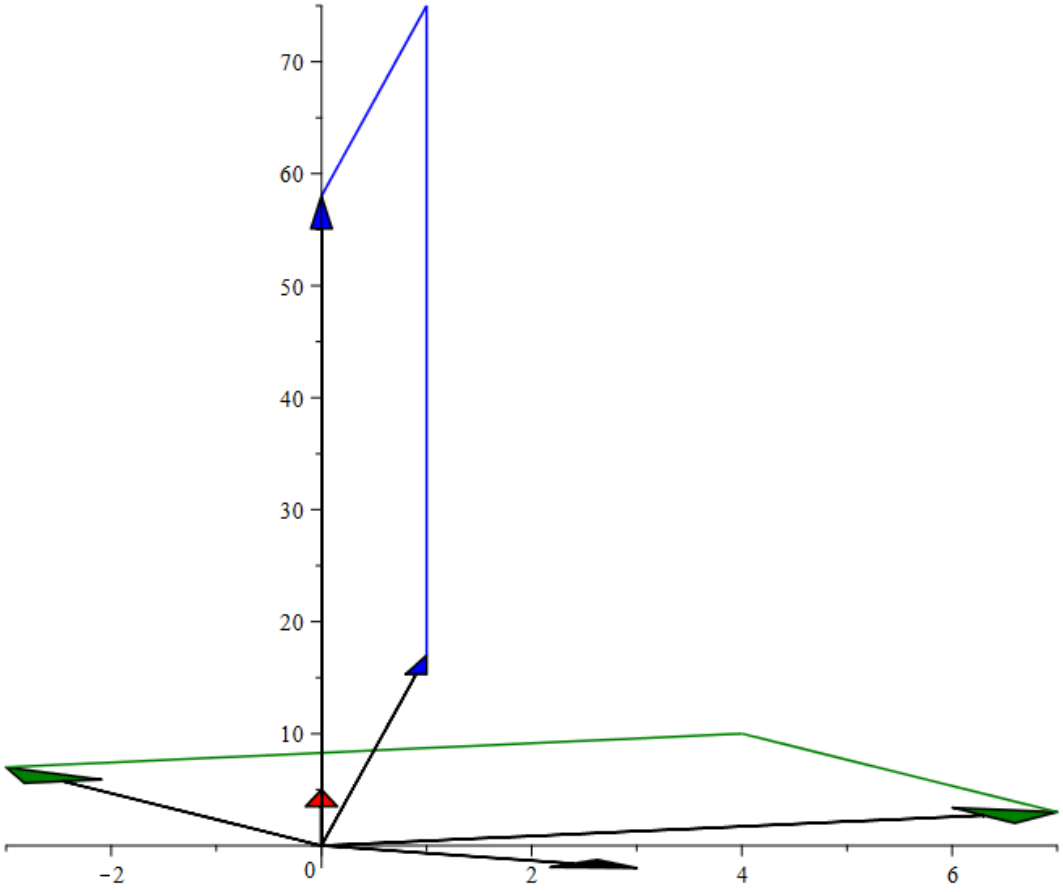

Figure 2.3: Good basis $B_{sk} = V$ (vectors with green heads), the parallelepiped formed by the good basis (with green lines), the bad basis formed by $HNF(V)$ (vectors with blue heads), and the parallelepiped of the bad basis (with blue lines).

Hadamard ratio for V is equal to 1 and computed using Maple according to (B.2) in Appendix B.

$v1 := Matrix([7, 3]):$
$v2 := Matrix([-3, 7]):$
$V := Matrix([[v1], [v2]])$

$$\begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$detV := \mathrm{abs}(Determinant(V))$

$$58$$

$$evalf\left(\left(\frac{detV}{norm(v1, 2) \cdot norm(v2, 2)}\right)^{\frac{1}{2}}\right)$$

$$1.$$

Hadamard ratio for $HNF(V)$ is approximately 0.24 and computed using Maple according to (B.2) in Appendix B.

$$hnfv1 := Matrix([1, 17]):$$
$$hnfv2 := Matrix([0, 58]):$$
$$HNF\_V := Matrix([[hnfv1], [hnfv2]])$$

$$\begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}$$

$$evalf\left(\left(\frac{detV}{norm(hnfv1, 2)\cdot norm(hnfv2, 2)}\right)^{\frac{1}{2}}\right)$$

0.2423262717

2- Encryption

To encrypt the bit $b = 1$, the vector $\vec{u}$ is selected as in (2.78).

$$\vec{u} = (1, -1). \tag{2.78}$$

The vector $\vec{a}$ is the vector encoding the message b using the noise $\vec{u}$ according to (2.73):

$$\vec{a} = 2\vec{u} + b \cdot \overrightarrow{e_1} = (3, -2). \tag{2.79}$$

The vector $\vec{a}$ encoding the message b in (2.79) can be seen in Figure 2.3 (the vector with black head). The ciphertext $\vec{c}$, is the vector encrypting the encoded message $\vec{a}$ according to (B.3) in Appendix B and (2.72).

$$\vec{c} = \vec{a}\ mod\ B = \vec{a} - \lfloor \vec{a} \times B^{-1} \rceil \times B =$$

$$[3 \quad -2] - \left[[3 \quad -2] \times \begin{bmatrix} 1 & -17/58 \\ 0 & 1/58 \end{bmatrix}\right] \times \begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}$$

$$= [3 \quad -2] - \left[\left[3 \quad -\frac{53}{58}\right]\right] \times \begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}$$

$$= [3 \quad -2] - [3 \quad -1] \times \begin{bmatrix} 1 & 17 \\ 0 & 58 \end{bmatrix}$$

$$= [3 \quad -2] - [3 \quad -7] = [0 \quad 5] \tag{2.80}$$

where $B^{-1} = \begin{bmatrix} 1 & -17/58 \\ 0 & 1/58 \end{bmatrix}$, the ciphertext $\vec{c}$ encrypts the message b in (2.80) can be found in Figure 2.3 (the vector with red hrad).

3- The decryption of ciphertext (2.80) is done in two steps as follows:

Step 1: The vector $\vec{a}$ is retrieved using $V^{-1} = \begin{bmatrix} 7/58 & -3/58 \\ 3/58 & 7/58 \end{bmatrix}$

$$\vec{a} = \vec{c}\ mod\ V = \vec{c} - \lfloor \vec{c} \times V^{-1} \rceil \times V \tag{2.81}$$

$$= [0 \quad 5] - \left\lfloor [0 \quad 5] \times \begin{bmatrix} \frac{7}{58} & -\frac{3}{58} \\ \frac{3}{58} & \frac{7}{58} \end{bmatrix} \right\rceil \times \begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$$= [0 \quad 5] - \left\lfloor [0 \quad 5] \times \begin{bmatrix} \frac{7}{58} & -\frac{3}{58} \\ \frac{3}{58} & \frac{7}{58} \end{bmatrix} \right\rceil \times \begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$$= [0 \quad 5] - \left\lfloor \begin{bmatrix} \frac{15}{58} & \frac{35}{58} \end{bmatrix} \right\rceil \times \begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$$= [0 \quad 5] - [0 \quad 1] \times \begin{bmatrix} 7 & 3 \\ -3 & 7 \end{bmatrix}$$

$$= [0 \quad 5] - [-3 \quad 7]$$

$$= [3 \quad -2]$$

Step 2: The message b is retrieved by applying modulo 2 operation to the first component of a according to (2.81). Thus, $b = 3\ mod\ 2 = 1$.

As shown in Sections 2.3, all known HE schemes w.r.t. to more than one operation type need noise control mechanism such as bootstrapping (Gentry, 2009b), i.e., "*the decryption procedure of the scheme is run homomorphically, using an encryption of the secret key that can be found in the public key, resulting in a new ciphertext that encrypts the same plaintext but has smaller noise* (Halevi & Shoup, 2021)". Using NCM each time the noise exceeds some threshold, makes it impractical to implement these FHE for arbitrary functions in cloud applications. According to (Sarkar et al., 2021, p. 133,134), the implementations of such encryption schemes remain unsuitable

for real-time applications yet due to the following reasons; long-time key generation, long-time circuit evaluation, and usage of memory, costly noise control mechanisms.

Experiments by (Gentry & Halevi, 2011) for implementing (Gentry, 2009b) showed that time required for key generation takes up to 2.2 hours, NCM consumes up to 31 minutes, and public-key size requires up to 2.25 GB. Therefore, in Chapter 7, we present the first FHE scheme not affected by noise growth and thus doesn't need NCM by construction.

2.7 Summary

A new homomorphic schemes classification is proposed, which addresses the challenge of grouping dissimilar HE into the same class by taking into account additional factors such as the necessity for CSCM, the underlying hard problem, and the number of encryption keys. The proposed classification increases the number of classes from four to at least 32. The proposed classification facilitates considering and studying the existent HE schemes in the literature, and better accommodate newly proposed HE schemes.

In order to outline challenges to be addressed in this thesis, a review of the HE schemes: RSA, NTRU, and HE1N has been presented from the literature, with the key shortcomings of each scheme emphasized. The problems that have been addressed are listed in the next section.

2.8 Problem Definition

The problems to be solved in the thesis are:

1. The problem of investigating RSA security. RSA is one of the first HE schemes, it supports homomorphism w.r.t. one operation that is multiplication.

It is used in CloudIoT for verifying node identities to prevent weak authentication (Pandey et al., 2020, p. 321; Yakubu et al., 2019, p. 226). RSA is used in the Internet Key Exchange protocol (IKE) that is designed specifically for use with IPsec Click or tap here to enter text.(Barker & Dang, n.d., p. 24) to provide peer authentication. RSA is also widely used in X.509, the standard defining the format of public-key certificates (*Rfc3279*, n.d.). Therefore, this thesis considers the problem of investigating RSA security.

2. Investigating NTRU Security. NTRU (Hoffstein et al., 1998) is a HE PKC w.r.t. two operations standardized as IEEE P1363.1 ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009)and announced as one of seven candidate algorithms in the third-round finalists of NIST Post Quantum Cryptosystem Standardization Process ((PQC Third Round Candidate Announcement | CSRC, n.d.). Therefore, this thesis considers the problem of investigating the security of NTRU.

3. The problem of proposing a new NTRU variant that is more efficient than NTRU and immune to LBRA attack. NTRU is prone to LBRA using LLL for low polynomial degrees. For this reason, NTRU polynomials are recommended to have degree $N > 400$ ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009) to meet a minimum-security level of 112-bit. Increasing polynomials degree increases the complexity of the encryption and decryption process. Therefore, the problem of proposing a new NTRU variant that is more efficient than NTRU and immune to LBRA attack is to be considered in this thesis.

4. The problem of investigating HE1N security is to be addressed in this thesis. HE1N (Dyer et al., 2019) is a symmetric HE schemes w.r.t two operations proposed as a practical scheme for cloud computing.

5. The problem of proposing an FHE scheme that is not affected by noise growth is to be considered in this thesis. As shown in Table 2.1, all known FHE schemes are L-FHE schemes need for NCM to be converted to FHE. For this reason,.all known FHE schemes are not practically used. Therefore, this thesis addresses the problem of proposing an FHE scheme that is not affected by noise growth.

Chapter 3

DESIGN OF CIPHERTEXT-ONLY ATTACK ON RSA (CLASS 1) USING LATTICE BASIS REDUCTION

We consider ciphertext-only attack (COA) on textbook RSA (Rivest et al., 1978), hereafter RSA, without preprocessing of the plaintext such as Optimal Asymmetric Encryption Padding (OAEP) used in RSA standard (Kaliski et al., 2016). Herein, we propose a new line of COA on RSA using LLL (Lenstra et al., 1982) algorithm to solve SVP in a 2-dimensional lattice. It is based on the first found herein opportunity of RSA encryption representation in terms of a 2-dimensional lattice. It doesn't require message broadcasting, prior knowledge of a part of a message/private key, or limitations on the size of public exponent *e*, contrary to all known approaches as shown in the last row of Table 3.1 but imposes constraints on the recoverable messages.

Table 3.1: Comparison between lattice-based COA and other known RSA attacks.

	Attack's Requirements		
Attack	Prior knowledge of a number of bits	A small value of exponent *e*	Broadcast messages
Coppersmith (Coppersmith, 1996a)	Yes	No	No
Boneh et al. (Boneh et al., 1998)Yes	No	No	No
Takayasu-Kunihiro ((Takayasu & Kunihiro, 2019)	Yes	No	No
Coppersmith (Coppersmith, 1996b)	No	Yes	No
Hastad (Hastad, 1988)	No	No	Yes
Bleichenbacher (Bleichenbacher, 1997)	No	No	Yes

Hastad (Hastad, 1986)	No	No	Yes
Simmons (Simmons, 1977)	No	No	Yes
DeLaurentis (Delaurentis, 1984)	No	No	Yes
Boneh (Boneh et al., 2000)	No	No	No
Bunder (Bunder et al., 2017)	No	Yes	No
Proposed Lattice-Based COA	No	No	No

Our COA attack computational complexity is $O(n^2)$, see Section 3.4. In our experiments, see Example 3.2, our attack on 2001 RSA 2050-bit messages took 45.775 seconds with about 0.1 success rate.

The rest of this chapter is organized as follows. Section 3.1, shows that RSA encryption forms a 2-dimensional RSA lattice. Section 3.2, shows that the plaintext message can be revealed as a component of the shortest vector in the RSA lattice. Section 3.3, proposes using LLL for COA on RSA by solving SVP in the RSA lattice. Section 3.4, evaluates the complexity of the proposed COA on RSA, and Section 3.5, conducts experiments for up to 8193-bit messages. Section 3.6 summarizes Chapter 3

3.1 Proposed Two-Dimensional RSA Lattice

RSA message recovery problem can be formulated as SVP in a 2-dimensional lattice, $L(V_1, V_2)$. From (3), we can see that:

$$c = m^j \cdot m^{e-j} mod\ N\ , j = 1..e-1 \tag{3.1}$$

and, hence,

$$m^j = \left(m^{e-j}\right)^{-1} \cdot c\ mod\ N. \tag{3.2}$$

From (3.2), we see that for any pair of integers, A and B, satisfying:

$$B\ =\ A \cdot c\ mod\ N. \tag{3.3}$$

(A, B) is likely to be $\left(\left(m^{e-j}\right)^{-1}, m^{j}\right)$, or (m^{-j}, m^{e-j}). Hence, equation (3.3) can be written as:

$$A \cdot c + N \cdot r = B, \tag{3.4}$$

where r is an integer. It forms a 2-dimensional RSA lattice,

$$A \cdot V_1 + r \cdot V_2 = (A, B), \tag{3.5}$$

where $V_1 = (1, c)$ and $V_2 = (0, N)$ are basis vectors, at least one of them having Euclidean norm of order $O(N)$, and determinant of the lattice equal to N.

3.2 Define RSA Message as the Shortest Vector in the RSA Lattice

According to Minkowski's Second theorem (B.5), vector (A, B) (3.5) likely is the shortest vector in the RSA lattice, if

$$\left|\left|A, B\right|\right| < 1.07\sqrt{N}. \tag{3.6}$$

Hence, our task is to find a pair of comparatively small, (A, B), satisfying (3.5) where $V_1 = (1, c)$ and $V_2 = (0, N)$ are known vectors. Then, (A, B) is likely to be $\left(\left(m^{e-j}\right)^{-1}, m^{j}\right)$, or $\left(m^{-j}, m^{e-j}\right)$. In our attack we adopt LLL to find the shortest vector in the 2-dimensional RSA lattice (3.5).

3.3 Design of LLL Attack on RSA Message as a Shortest Vector in the RSA Lattice

We want to find the shortest vector w from $L(V_1, V_2)$ using LLL that might disclose

$$(A, B) = \left(\left(m^{e-j}\right)^{-1}, m^{j}\right) \tag{3.7}$$

if $\left|\left|\left(\left(m^{e-j}\right)^{-1}, m^{j}\right)\right|\right|$ from (3.7) is of the order of $O(\sqrt{N})$ meeting (3.6). In our experiments, we used the LLL algorithm implemented in Maple 2016.2. Example 3.1 shows LLL attack on Example A.1 message, Appendix A.

Example 3.1: LLL attack on 40-bit RSA message from Example A.1 in Appendix A

The ciphertext from Example A.1, $c = 480808351840$, and modulus $N = 1099559862701$. Hence, $V_1 = (1, 480808351840)$, and $V2 = (0, 1099559862701)$. LLL attack with $V_1 = (1, 480808351840)$, $V_2 = (0, 1099559862701)$, defined in (3.5) terminates in 15 milliseconds using Maple, obtaining the shortest vector (see Figure 3.1) given in (3.8):

$$v_1 = (82493, 986648). \tag{3.8}$$

```
with(IntegerRelations);
                    [LLL, LinearDependency, PSLQ]
e := 2^16 + 1
                              65537
p := 2^20 + 33
                             1048609
q := 2^20 + 13
                             1048589
m := 986648
                              986648
N := p·q
                          1099559862701
c := m^e mod N
                          480808351840
V1 := [1, c]
                       [1, 480808351840]
V2 := [0, N]
                       [0, 1099559862701]
v := LLL([V1, V2])
          [[82493, 986648], [-1136417, -262855]]
v[1]
                        [82493, 986648]
time(LLL([V1, V2]));
                              0.015
```

Figure 3.1: LLL attack on RSA message in Example A.1 using Maple 2016.2

We also run Example 3.1 in C using NTL (*NTL: A Library for Doing Number Theory*, n.d.) and found that the LLL attack terminates in 4×10^{-5} seconds. Thus, we see that our attack, both in Maple and C, takes less time than attacks mentioned in Subsection 2.2.3. LLL attack succeeds to retrieve message since it is a component of the shortest vector in the lattice,

$$\left|\left|(m^{e-1})^{-1}, m\right|\right| \approx 990090.6 < 1.07\sqrt{N} \approx 1124497.2.$$

The complexity of the LLL algorithm is presented in Section 3.4.

3.4 Complexity of LLL Lattice Basis Reduction Algorithm

Lenstra, Lenstra, and Lovasz (Lenstra et al., 1982) state that for n-dimensional lattices with integer input basis vectors of bounded length N, the LLL algorithm terminates after at most $O(n^2 \log N)$ iterations.

3.5 Experiments on RSA Cracking for Up To 8193-Bit Messages

We have conducted experiments using Maple 2016.2 in Windows 8.1 on Lenovo laptop with Intel i5-6200U CPU 2.30 GHz, 8 GB RAM, for RSA with p, q values specified in Table 3.2 with sizes of

$$N = p \cdot q, \tag{3.9}$$

from 40 to 8193 bits more than twice exceeding recommended RSA key size, 4047 bits, for the 2050 year according to the requirements of (Lenstra & Verheul, 2000). Values of p, q are defined as integer expressions (see Table 3.2). Experiments in Table 3.2 conducted using Digits=10 and C = 0 in Maple.

Table 3.2: Number of cracked messages under different parameter settings.

1	2	3	4	5	6	7	8
Pair (p, q)#	p	q	Bit size of N	(a, b) from (3.13) for which RSA was cracked, k = 1	δ_{min}	δ_{max}	Number of RSA cracks
0	$2^{20} + 33$	$2^{20} + 13$	40	$(8, 1)$, $(4, \pm 1)$, $(2, \pm 1)$	0.025	0.508625	153
1	$2^{130} - 5$	$2^{131} + 39$	261	$(14, -1)$, $(4, 2)$, $(2, \pm 1)$	0.01	0.5010325	58
2	$3 \times 2^{250} + 17$	$\left(2^{129} - 1\right)^2 - 2$	509	$(2, \pm 1)$, $(4, -1)$, $(22, \pm 1)$	0.01	0.5007125	59

3	$3 \times 2^{512} + 349$	$3 \times 2^{512} - 511$	1026	$(4,1)$, $(8,-1)$, $(2,\pm 1)$	0.01	0.5007065	85
4	$3 \times 2^{1024} + 515$	$3 \times 2^{1024} - 1717$	2050	$(20,6)$, $(4,2)$, $(2,\pm 1)$	0.01	0.5005	64
5	$3 \times 2^{2048} + 595$	$3 \times 2^{2048} - 1105$	4098	$(26,5)$, $(4,2)$, $(2,\pm 1)$	0.001	0.5007	68
6	$3 \times 2^{4096} + 1075$	$2^{4096} - 2549$	8193	$(28,4)$, $(4,2)$, $(2,\pm 1)$, $(14,-1)$	0.003	0.50003	66

Note that the prime values (p, q) used in Rows 1, 2 of Table 3.2 are strong according to (Rivest et al., 1978, p. 124), since $p-1, q-1$ have large primes as their factors, that is confirmed by the following Maple code:

```
p := 2^130 - 5; isprime(p);
        1361129467683753853853498429727072845819
                                true                                (1)
q := 2^131 + 39; isprime(q);
        272225893536750770770699685945414569168 7
                                true                                (2)
ifactor(q-1), ifactor(p-1);
(2) (7) (139) (419) (33386629146471529723869008080391 89),          (3)
    (2) (23) (32985101) (89706473951992278723018299378 3)

p1 := 3*2^250 + 17; isprime(p1); q1 := (2^129-1)^2-2; isprime(q1);
54277541829991966604798899222822456806220305312019014393495 74
    250370927951889
                                true
46316835694926478169428394003475163141171880919487850230397 68
    37601925445713919
                                true                                (4)

ifactor(q1 - 1)
(2) (11) (13)                                                       (5)
    (
    16194697795428838520779158742473833266144014307513234346292896419 6
    4774285713)
ifactor(p1 - 1)
(2)^4 (103) (2639809)                                               (6)
    (
    12476433558590379748972980092908838430300119065326421617703622351 5
    9)
```

It can be checked that (p, q) values in rows 1, 2, and 6 of Table 3.2 have large $round(N^{0.5} - \lfloor p^{0.5} \rfloor \cdot \lfloor q^{0.5} \rfloor)$ values precluding attack. In our experiments, messages are defined via a parameter,

$$\delta \in (0,1), \tag{3.10}$$

as follows,

$$m = int(N^{\delta}) + ii, ii \in -C, \cdots, C, C \geq 0, \tag{3.11}$$

where $C \geq 0$ is an integer and $int(\,)$ returns the integer part of its input. Calculations on the float-point numbers are done with an accuracy of 10, 15, 100, 200, 600, 800, and 1600 digits:

```
> Digits := 1600;#600;#200;#10;#15
                        Digits := 1600
```

We try vectors

$$v(j) = \left(\left(m^{e-j} \right)^{-1}, m^{j} \right) \tag{3.12}$$

meeting the following two-dimensional lattice equation

$$v(j)_1 \cdot V_1 + r \cdot V_2 = v(j) \tag{3.13}$$

with

$$V_1 = \begin{pmatrix} 1 \\ c \end{pmatrix}, V_2 = \begin{pmatrix} 0 \\ N \end{pmatrix} \tag{3.14}$$

For $j = 1, \cdots, 100$, according to (3.5), by the following code:

Code 3.1: Maple code for RSA cracking using LLL with $j \in \{1, \cdots, 100\}$. Initial conditions for the code are defined in Code 3.3 and Example 3.2. It tries cracking 2001 RSA messages in the range $m0 - 1000 \ldots m0 + 1000$, where $m0$ is defined in its first line as trunc(N^{δ}).

```
st := time( ) : bnd := 1.07·N^0.5 : lb := [ ] : gb := [ ] : m0 := trunc(N^δ) :
 for ii from  -1000 to 1000  do#−992 do#−1000 to 1e3 do
 m := m0 + ii;
p0 := expmN(m, e − 1, N); igcdex(N, p0,'z','me1');
 c := p0·m mod N;
nrm := (me1^2 + m^2)^0.5;
 V := [[1, c], [0, N]] ; VR := LLL(V,'integer');
 for j to 1e2 do
  if abs(VR[1, 1]) = m^j or abs(VR[1, 2]) = m^j or abs(VR[2, 1]) = m^j or
abs(VR[2, 2]) = m^j then
    if nrm < bnd then lb := [op(lb), [ii, j]] else gb := [op(gb), [ii, j]]
end if
  end if
 end do:
 end do: ft := time( ) : tot := ft − st;
```

In Code 3.1, with C = 1000, we check the both returned by LLL vectors and each their component on equality to m^j. Exponentiation function and LLL used in Code 3.1 are introduced in Code 3.2 as follows:

Code 3.2: Maple code introducing exponentiation function and LLL.

```
expmB :=proc(m :: integer, e :: integer, N :: integer) :: integer;
    local p1, m0, p0;
   description "Exponentiate m power e mod N";
p0 := 1 : p1 := e : m0 := m :
while(p1 ≠ 0) do
if p1 mod 2 ≠ 0 then p0 := p0·m0 mod N end if;
m0 := m0^2 mod N;
p1 := trunc(p1/2);
end do;
return p0;
end proc:
with(IntegerRelations)
```

RSA was successfully cracked under conditions (3.16)-(3.18) on the encryption key, e, defined via Euler totient function,

$$\phi(N) = (p-1)\cdot(q-1), \tag{3.15}$$

in a general form

$$e = k\cdot\frac{\phi(N)}{a} - b, \tag{3.16}$$

such that

$$\gcd\left(e,\phi(N)\right) = 1, \tag{3.17}$$

$$\phi(N)\ mod\ a = 0. \tag{3.18}$$

It is implemented in Maple by the following Code 3, for Digits=1600:

Code 3.3: Maple implementation of RSA encryption key, *e*, calculation according to (3.16)-(3.18), for *N* of 2050 bit size from Table 3.2.

```
N := p*q : φ := (p-1)*(q-1) :
a := 20 :
b := 6;
k := 9;
e := k·φ/a - b : evalf(e);#φ/4 - 9 : evalf(e)
# e:=trunc(N^α) - 1 :
igcdex(φ, e, 'z','d'); d : z : e·d + z·φ; d := d mod φ :
                                  b := 6
                                  k := 9
1.308838745888095795678952505891133054397986158123477103301278989814
   748533124149581174486657166677995279393077417336869789511842145 5
   948823043510701716511554663855505333756903417158830088907809103 2
   524033435455473651521008485502610012455444282070828490405855665 7
   007470354236946823607413165248605872164519180610715560080876359 5
   916145288217473842625190153478578909400298185900651196482620799 8
   796404188261258708274011438494716807758759676431935868887712960 7
   615721935715228459250503024620812852585221286704782931570671326 9
   124996740616985739047693707450705081662992364166858177801073960 3
   286289945125318304513028953577451705891 10^617

                                  1
                                  1
```

For the example of data shown in Code 3.2, $e \approx \frac{N}{2}, d \approx \frac{N}{10}$, thus attacks described in Section 2.2.3 are not applicable. We try finding a range of the parameter,

$$\delta \in [\delta_{min}, \delta_{max}], \tag{3.19}$$

or a set of values, $\{\delta_{min}, \delta_{max}\}$, for which our method successfully cracks RSA (see Table 3.2, columns 7, 8). In Table 3.2, columns 5, 6, pairs (a, b), for which RSA was successfully cracked, and number of successful cracks are given (for $C = 0$ in (3.11)). We found that for all successful cracks,

$$j = |b|, \tag{3.20}$$

holds, where *j,b* are from(3.12), (3.13), and (3.16), respectively, i.e., the power of the plaintext message, *m*, revealed by our attack on RSA, always is equal to $|b|$ from (3.16). Thus, in the experiments, we find two conditions, (3.18) and (3.20), holding that need explanation. Also, the results of all our experiments show that condition (3.21) holds

$$\delta_{max} \cdot |b| \approx 0.501. \tag{3.21}$$

To verify (3.21), we have conducted a special massive investigation of its validity for (p, q) pair from Table 3.2, row 4, results of which are given in Table 3.3, and confirm its validity. Hence, we need explaining (3.18), (3.20), and (3.21). Experiments in Table 3.3 are conducted with Digits=600, $p := 3 \cdot 2^{1024} + 515$, $q := 3 \cdot 2^{1024} - 1717$, (3.16)-(3.18) hold, $k = 1$, $C = 1000$, and δ_{max} is from (3.19).

Table 3.3: Results of experiments on RSA cracking with $N = 2050$

a	b	**Number of cracks**	δ_{max}	$\delta_{max} \cdot \|b\|$
20	6	9205	0.0835	0.501
20	10	4987	0.050107	0.50107
20	-4	2082	0.12521	0.50084
20	-6	1642	0.038348	0.50088
20	-8	1913	0.062615	0.50092
20	-14	1336	0.035769	0.500766
5	-1	5626	0.501	0.501
5	-25	2896	0.020045	0.501125
4	2	21599	0.25066	0.50132
4	6	22937	0.083528	0.501168

10	13	6469	0.0385503	0.501154
Total cracks: 80692			**Average $\boldsymbol{\delta_{max}} \cdot |\boldsymbol{b}|$:** 0.501022	

Explanation of (3.18): Consider (3.10)-(3.12) for $C = 0$, (3.15), and (3.16). Then, RSA ciphertext, c, is defined as follows:

$$c = m^e mod\ N = m^{k \cdot \frac{\phi(N)}{a} - b} mod\ N. \quad (3.22)$$

Experiments show that with high probability, ranging from 0.1 to 0.5, (3.23) holds:

$$m^{\frac{k\phi(N)}{a}} mod\ N = 1. \quad (3.23)$$

Note that due to Euler's theorem (Stallings, 2014),

$$m^{k\phi(N)} mod\ N = 1, \quad (3.24)$$

and the left-hand side (LHS) of (3.23) is the a-th root of unity from LHS of (3.24), which is highly likely to be also unity. The probability of our COA on RSA success estimate is illustrated in Example 3.2.

Example 3.2: Conducting calculations by Code 3.1 in Maple 2016.2, with Digits=1600, $q = 3 \cdot 2^{1024} - 1717, p = 3 \cdot 2^{1024} + 515, \delta = 0{:}071435$ considering 2001 numbers, $m = \lfloor N^{\delta} \rfloor + ii$, $ii \in [-C, \cdots, C], C = 1000$, we find 216 cases when (3.23) holds, in particular, for $ii = -998, -992, -988$, etc. Respective Maple output is shown in Figure 3.2. Thus, the probability of (3.24) holding, and thus our attack takes 45.775 seconds, its success probability under conditions (3.16)-(3.18), may be estimated as 216/2001=0.1079, and (3.17) is explained. Now, we explain (3.19) and (3.20).

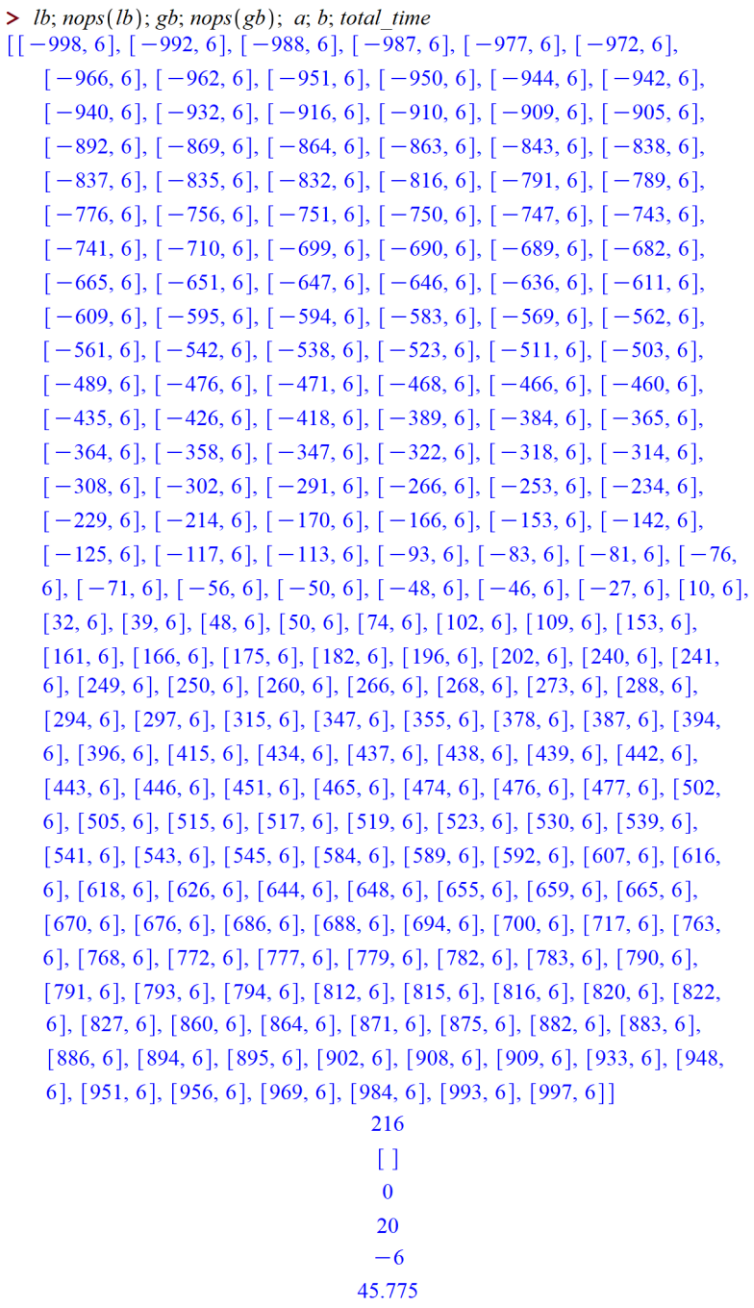

Figure 3.2: Screenshot of Maple implementation of Code 3.1 using parameter settings in Example 3.2

Explanation of (3.19) and (3.20): Our method of cracking of RSA ciphertext is as follows (recall (3.1)-(3.5), (3.12), (3.13)). Rewrite (3.22):

$$c = m^{e-j} \cdot m^j \bmod N, 0 < j < e. \tag{3.25}$$

From (3.15), we get

$$c\left(m^{e-j}\right)^{-1} = m^j \bmod N. \tag{3.26}$$

Reminding (3.12), from (3.26), we arrive at (3.13). Applying LLL algorithm to the lattice defined by (3.14), we obtain the shortest vector, $\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}$, of the lattice such that $\begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = v(j)$, if the norm of $v(j)$ meets Minkowski's Second theorem:

$$\left\|\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}\right\| \leq ||v(j)|| = \sqrt{v(j)_1^2 + v(j)_2^2} \leq \sqrt{\gamma_2 \cdot N} = \sqrt{\frac{2}{\sqrt{3}} \cdot N} \tag{3.27}$$

where $\gamma_2 \approx 1.1547$ is Hermite's constant for the 2-dimensional lattice. To meet (3.27), from (3.12), we have:

$$\sqrt{(m^{-e+j})^2 + m^{j^2}} \leq \sqrt{\frac{2}{\sqrt{3}} \cdot N} \tag{3.28}$$

From, (3.11) with *C* = 0, (3.16), (3.23), (3.28), we have:

$$\sqrt{(m^{b+j})^2 + m^{j^2}} = \sqrt{\lfloor N^\delta \rfloor^{b+j} + \lfloor N^\delta \rfloor^{j^2}} \leq \sqrt{2 \cdot \frac{N}{\sqrt{3}}} \approx N^{0.50005} \tag{3.29}$$

From (3.29), we have two cases:

Case 1: $b \geq 0$. Let $j = 0$ in (3.29). Then, $v(j) = \begin{pmatrix} m^b \\ 1 \end{pmatrix}$, and we have:

$$\sqrt{\lfloor N^\delta \rfloor^{b^2} + 1} \approx N^{b \cdot \delta} \leq N^{0.5005}, \tag{3.30}$$

and thus,

$$b \cdot \delta \leq 0.50005 \tag{3.31}$$

Case 2: $b < 0$. Let $j = -b = |b|$. Then, $v(j) = \begin{pmatrix} 1 \\ m^b \end{pmatrix}$, and we have

$$\sqrt{\lfloor N^\delta \rfloor^{|b|^2} + 1} \approx N^{|b| \cdot \delta} \leq N^{0.5005}, \tag{3.32}$$

and then,

$$|b| \cdot \delta \leq 0.50005. \tag{3.33}$$

Thus, from (3.30), (3.32), we may have RSA cracks the form

$$|v(j)| = \begin{pmatrix} m^b \\ 1 \end{pmatrix} \text{ or } |v(j)| = \begin{pmatrix} 1 \\ m^b \end{pmatrix}, \tag{3.34}$$

that have been observed in all our experimental results shown in Table 3.2 and Table 3.3. Example 3.2 confirms that (3.34) holds in a particular experiment as in all other ones.

Example 3.3: Maple output for RSA cracking with $k = 9$, $a = 20$, $b = \pm 6$, $d = 0{:}071435$, showing that (3.23) holds, and values found by LLL in $VR[1,1..2]$, see Code 3.1, meet (3.34).

$$expmN\left(m, 9\cdot\frac{\varphi}{a}, N\right), evalf\left(m^6\right), VR[1,1], VR[1,2], a, b$$

1,

4.47821720597401582614759033248188734126656659387525000928318457040911054
07224490985585125770487178156312236574564346740572953361851540294968067 06
83748244618691360073711122435228305975699105263199374043948986128389842 14
96734958262021002560149182646309877686853504 64 10^{264}, 1,
44782172059740158261475903324818873412665665938752500092831845704091105 45
07224490985585125770487178156312236574564346740572953361851540294968067 06
83748244618691360073711122435228305975699105263199374043948986128389842 14
96734958262021002560149182646309877686853504 64, 20, −6

> $expmN\left(m, 9\cdot\frac{\varphi}{a}, N\right), evalf\left(m^6\right), VR[1,1], VR[1,2], a, b$

1,

4.47821720597401582614759033248188734126656659387525000928318457040911054 5
07224490985585125770487178156312236574564346740572953361851540294968067 06
83748244618691360073711122435228305975699105263199374043948986128389842 14
96734958262021002560149182646309877686853504 64 10^{264},
−44782172059740158261475903324818873412665665938752500092831845704091105 4
50722449098558512577048717815631223657456434674057295336185154029496806 70
68374824461869136007371112243522830597569910526319937404394898612838984 21
49673495826202100256014918264630987768685350464, −1, 20, 6

Also, the range for δ defined by (3.31), (3.33) is confirmed by our experiments. From Table 3.3, the last row, we see that (3.33) holds on average with accuracy 0.00097=0.50102-0.50005. Table 3.3 contains the number of RSA successful cracks for different values of a,b, maximal δ_{max} from (3.19) and LHS of (3.33). Thus, (3.34) explains (3.20), and (3.34) explains (3.21). To find the relation between a and number of RSA successfully cracked messages, we run Code 3.1 with p, q from rows 3-6 of Table 3.2, $\delta \in 0.01, \ldots, 0.52$ yields to launch 104,052 attacks on each (p, q, a) value. Figure 3.3 shows an inverse proportion between value of a and number of successful cracks. Thus, decreasing the public key leads to a decrease in the success rate of our attack.

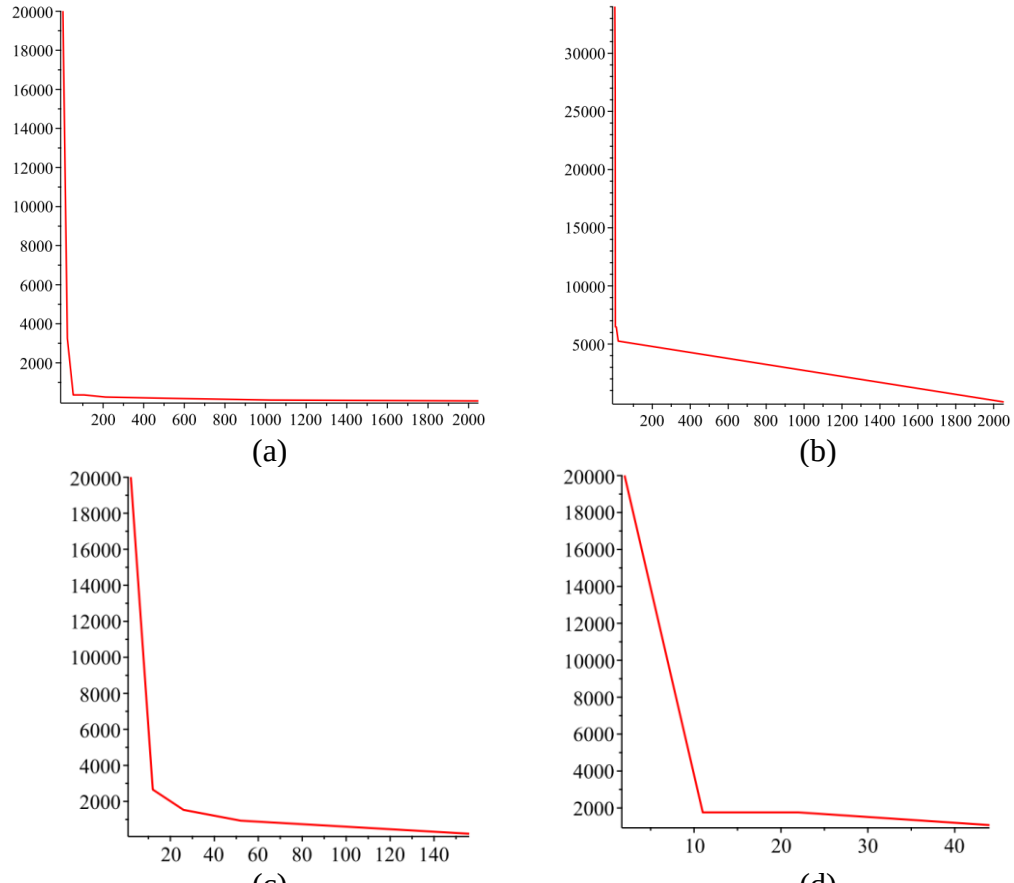

Figure 3.3: Inverse relation between the value of parameter a (horizontal axis) in (3.16) and number of successful RSA message cracks (vertical axis) out of 104,052 message attacks.

In Figure 3.3, (a) shows 20010 message cracks at a =2 and drops to 51 message cracks at a = 2048 out of 104,052 message attacks. (b) shows 34017 message cracks at a = 2 and drops to 18 message cracks at a=2053 out of 104,052 message attacks. (c) shows 20010 message cracks at a = 2 and drops to 0 message cracks at a = 33739 out of 104,052 message attacks. (d) shows 20010 message cracks at a = 2 and drops to 199 message cracks at a = 222 out of 104,052 message attacks

3.6 Summary

In this chapter, we show that RSA-encrypted message considered as a component of the shortest vector of the RSA lattice can be revealed by LLL attack. LLL attack runs in time quadratic in the bit number of modulo N (see Section 3.4). LLL attack targets messages meeting (3.2)-(3.6) being the shortest vector in the RSA lattice. Our attack works in the conditions discussed in Section 3.2 in which known attacks can't work, and it does not impose any other requirements, such as the need for very small public exponent, e, part of the plaintext to be known in advance, or a message broadcasting to sufficiently many participants, each holding a different modulus with a known affine transformation, or using common modulus as other attacks do (Boneh et al., 1998; Coppersmith, 1996a, 1996b; Hastad, 1988, 1986; Takayasu & Kunihiro, 2019). Our attack shows significant speed (15 milliseconds using Mupad, and 4×10^{-5} seconds using NTL library for Example 3.1) in recovering a 40- bit message in comparison to our implementation for Boneh MITM attack (Boneh et al., 2000) where 2.202 seconds are needed to recover the same length message (2 seconds for pre-computation step, and 0.202 seconds for searching step using NTL library). Additionally, we have conducted experiments with the proposed method for N with bit sizes up to 8193 in Maple 2016.2, with results presented in Table 3.2, Table 3.3, in which thousands of successful RSA cracks were conducted using Code 3.1 run-time of which in the

conditions of Example 3.2 for 2001 RSA 2050-bit messages cracking is about 45 seconds. The cracks were made for large public key values meeting (3.16)-(3.18) for which truth of (3.18), (3.20), (3.21) was discovered. Based on these findings, for RSA not to be susceptible to the attack proposed herein, it is recommended RSA public keys be selected such that (3.16)-(3.18) are not satisfied.

RSA is HE cryptosystem that is widely used in the Interned to provide authentication, and many applications exploit its homomorphic multiplicative feature, such as digital signature in electronic voting. Therefore, it is crucial to select RSA public keys such that (3.16)-(3.18) are not satisfied in order to prevent the proposed COA attack.

Chapter 4

SECURITY ANALYSIS OF NTRU (CLASS 3)

In this chapter, the security of NTRU is analyzed through two sections; Section 4.1 presents the NTRU flaw that allows in some cases revealing encrypted messages without the need for the private key, non-negligibility of the attack is derived, and recommended settings are proposed to avoid this attack. Section 4.2 presents experiments that examine the efficiency of $IN - Lattice$ Attack (Z. Yang et al., 2018b), and Section 4.3 summarizes results of Chapter 4

4.1 Design of NTRU Modulo p Flaw Attack

In this section, we prove that for some parameters NTRU has the modulo p flaw (Chefranov & Ibrahim, 2016; Ibrahim & Chefranov, 2016), so NTRU-encrypted plaintext can be disclosed just by applying modulo p operation to the ciphertext without the need of using any of NTRU secret keys. We provide also statistical estimates of the probability of having NTRU modulo p flaw cases for different values of N, where N is the order of polynomial ring used in NTRU. The probabilities show that NTRU modulo p flaw may take place rather often. NTRU amendment to withstand the flaw is proposed. The rest of this section is organized as follows. In Subsection 4.1.1, the NTRU modulo p flaw is shown by example; explanations are given for the example. In Subsection 4.1.3, we present statistics of cases when NTRU has modulo p flaw for different N values.

4.1.1 NTRU Modulo p Flaw Attack

Attack on NTRU using modulo p flaw may be conducted as follows.

Step 1: Center-lift ciphertext, e, (2.20) w.r.t q

Step 2: Apply modulo p operation to center-lifted ciphertext from Step 1.

A numerical example of NTRU mod p flaw is presented in Example 4.1 the explanation of the example follows.

Example 4.1: Example of NTRU modulo p flaw

Let according to (2.13),

$$N = 5; d = 1; p = 3; q = 32 > (6d + 1)p = 21, \tag{4.1}$$

Let according to (2.14),

$$f(x) = x^3 + x^2 - 1 = [-1,0,1,1,0] \tag{4.2}$$

Then, according to (2.15),

$$F_q = x^4 + x - 1 \; mod \; 32, F_p = x^4 + x - 1 \; mod \; 3 \tag{4.3}$$

Let according to (2.16), (2.18) and (2.19), (x) ,$r(x)$, and $m(x)$ are selected as,

$$r(x) = x - 1, \;\; g(x) = x - 1, \;\; m(x) = x^2 + x + 1 \in R_p. \tag{4.4}$$

The public key, h, according to (2.17) is:

$$h = F_q \cdot g \; mod \; q = (x^4 + x - 1)(x - 1) = -x^4 + x^2 - 2x + 2 \in R_q \tag{4.5}$$

Ciphertext according to (2.20),(4.4), and (4.5), is:

$$F_{h,p}^{NTRU}(m, r) = e = p \cdot r \cdot h + m \; mod \; q$$

$$= 3x^4 + 3x^3 + 24x^2 + 13x + 24 \in R_q. \tag{4.6}$$

Attack on NTRU using modulo p flaw may be conducted as follows.

Step 1. Center-lift ciphertext,e, (4.6) w.r.t $q = 32$:

Step 2. Apply modulo p operation to center-lifted ciphertext from Step 1.

$$e = 3x^4 + 3x^3 + 24x^2 + 13x + 24$$

$$3x^4 + 3x^3 - 8x^2 + 13x - 8 \ mod\, q. \tag{4.7}$$

Then according to Step 2, we apply modulo p operation directly to the center-lifted ciphertext (4.7), we also disclose the original plaintext,$m(x)$, from (4.4), as follows

$$m = e\, mod\, p = (3x^4 + 3x^3 - 8x^2 + 13x - 8)\, mod\, 3$$

$$= x^2 + x + 1 \in R_p. \tag{4.8}$$

Comparing (4.8) and (4.4), we see that actually, the plaintext is restored without any key, by knowledge of the public value of public parameter p only. Thus, the example represents NTRU flaw that we call "modulo p flaw".

4.1.2 Explanation of Example 4.1

The reason for the NTRU modulo p flaw in Example 4.1 is that in the encryption (2.20) it might happen that the polynomial, A, used for hiding the plaintext, $m(x)$, from (2.18),

$$A = p \cdot r \cdot h = [\alpha_0, \ldots, \alpha_{N-1}]. \tag{4.9}$$

has all its coefficients by an absolute value less than q considering h center-lifted. This condition can be written as follows:

$$\forall i \in \{0, \ldots, N-1\},\ |\alpha_i| < q, \tag{4.10}$$

Where α_i is the i-th coefficient of polynomial A in (4.9). We can see that (4.10) holds in Example 1 (see (4.6)). In such a case, modulo q operation used in (4.6), preserves A being a multiple of p that can be eliminated from (4.6) just by modulo p operation applied to the ciphertext, e, as we exactly made in (4.8) after center-lifting e in (4.7).

For the NTRU modulo p flaw realization, we need to find such inverse of (2.14) that the products (2.17), (4.9) used in (2.20), have coefficients by an absolute value less than q (see (4.10)). Hence, we need finding dependence of the products' coefficients on the coefficients of (2.14). It is done in the next Subsection 4.1.2.1. Then, in Subsection 4.1.2.1, we find such polynomial (2.14) that the product (4.9) likely has coefficients by an absolute value less than q.

4.1.2.1 Finding Inverse of the Polynomial *f(x)* Modulo (x^N-1)

Consider the finding of an inverse, $f^{-1}(x) = [b_0, b_1, \ldots, b_{N-1}]$, of (2.14) in R.

$$f(x) \cdot f^{-1}(x) \bmod x^N - 1 = 1 \tag{4.11}$$

From (4.11),

$$f(x) \cdot f^{-1}(x) = c(x) \cdot (x^N - 1) + 1 \tag{4.12}$$

$$c(x) = \sum_{i=0}^{N-2} c_i x^i \tag{4.13}$$

From (2.14), (4.12), and (4.13):

$$\sum_{i=0}^{2N-2} \left(\sum_{\substack{j+k=i, \\ 0 \le j,k < N}} f_j b_k \right) x^i = \sum_{i=N}^{2N-2} c_{i-N} x^i - \sum_{i=0}^{N-2} c_i x^i + 1 \tag{4.14}$$

Equating coefficients near respective powers, we get from (4.14) the following system of linear algebraic equations w.r.t unknowns, $b_0, .., b_{N-1}, c_0, .., c_{N-2}$,

$$\sum_{j=i-N+1}^{N-1} f_j b_{i-j} = c_{i-N}, \text{for } i = N, \ldots, 2N-2 \tag{4.15}$$

$$\sum_{j=0}^{N-1} f_j b_{N-1-j} = 0 \tag{4.16}$$

$$\sum_{j=0}^{i} f_j b_{i-j} = -c_i, i = 1, \ldots, N-2 \tag{4.17}$$

$$1 - f_0 b_0 = c_0 \tag{4.18}$$

Preserving $b_0, .., b_{N-1}$ only, from (4.15) -(4.18), we get

$$\sum_{j=0}^{i-N} f_{i-N-j} b_j + \sum_{j=i-N+1}^{N-1} f_{i-j} b_j = 0, \text{for } i = 2N-2, \ldots, N+1 \tag{4.19}$$

$$f_0 b_0 + \sum_{j=1}^{N-1} f_{N-j} b_j = 1 \tag{4.20}$$

$$\sum_{j=0}^{N-1} f_{N-1-j} b_j = 0 \tag{4.21}$$

For $N = 5$, the matrix of coefficients in (4.19)-(4.20) is as follows

$$\Delta = \begin{bmatrix} f_3 & f_2 & f_1 & f_0 & f_4 \\ f_2 & f_1 & f_0 & f_4 & f_3 \\ f_1 & f_0 & f_4 & f_3 & f_2 \\ f_0 & f_4 & f_3 & f_2 & f_1 \\ f_4 & f_3 & f_2 & f_1 & f_0 \end{bmatrix} \tag{4.22}$$

Determinant of (4.22), $det(\Delta)$, calculated using Maple 2016, is as follows (Figure 4.1):

```
with(LinearAlgebra) :
delta := Matrix([[f3, f2, f1, f0, f4], [f2, f1, f0, f4, f3], [f1, f0, f4, f3, f2], [f0, f4, f3, f2, f1], [f4, f3, f2, f1, f0]])
```

$$\begin{bmatrix} f3 & f2 & f1 & f0 & f4 \\ f2 & f1 & f0 & f4 & f3 \\ f1 & f0 & f4 & f3 & f2 \\ f0 & f4 & f3 & f2 & f1 \\ f4 & f3 & f2 & f1 & f0 \end{bmatrix} \tag{1}$$

```
DetDelta := Determinant(delta, method = multivar)
```

$$f0^5 - 5\,f0^3\,f1\,f4 - 5\,f0^3\,f2\,f3 + 5\,f0^2\,f1^2\,f3 + 5\,f0^2\,f1\,f2^2 + 5\,f0^2\,f2\,f4^2 + 5\,f0^2\,f3^2\,f4 - 5\,f0\,f1^3\,f2 + 5\,f0\,f1^2\,f4^2 - 5\,f0\,f1\,f2\,f3\,f4 - 5\,f0\,f1\,f3^3 - 5\,f0\,f2^3\,f4 + 5\,f0\,f2^2\,f3^2 - 5\,f0\,f3\,f4^3 + f1^5 - 5\,f1^3\,f3\,f4 + 5\,f1^2\,f2^2\,f4 + 5\,f1^2\,f2\,f3^2 - 5\,f1\,f2^3\,f3 - 5\,f1\,f2\,f4^3 + 5\,f1\,f3^2\,f4^2 + f2^5 + 5\,f2^2\,f3\,f4^2 - 5\,f2\,f3^3\,f4 + f3^5 + f4^5 \tag{2}$$

Figure 4.1: Definition of the matrix (4.22), and its determinant, in Maple 2016

Right hand side, RHS, of equations (4.19) -(4.22), for $N = 5$ is as follows

$$RHS = (0,0,0,1,0) \tag{4.23}$$

Using Cramer's rule (Strang, 2016), find

$$b_i = \frac{det(\Delta_i)}{det(\Delta)}, \text{for } i = 0, \ldots, N-1, \tag{4.24}$$

where the matrix Δ_i is the matrix Δ with column i replaced by $RHS = (0,0,0,1,0)$, (4.24) is the right side of (4.27).

Division in (4.24) is made modulo p or q to find F_p or F_q from (2.15) respectively. For the correctness of the division in (4.24), determinant in the denominator shall have multiplicative inverse modulo p and q, and shall be co-prime to them. For arbitrary determinants, their inverses may be rather large integers resulting in large coefficients b_i in (4.24), hence, leading to large coefficients in h (2.17), and, thus, to violation of (4.10). To minimize the coefficients, we need the absolute value of the determinant value,$det(\Delta)$ (see (4.22)), equal to 1. Such a case is considered in the next Subsection 4.1.2.2 and was used in Example 4.1.

4.1.2.2 Getting |det(Δ)|=1

For the polynomial (4.2) used in Example 4.1, from (4.22) and Figure 4.1, we have,

$$|det(\Delta)| = 1; \tag{4.25}$$

$$det(\Delta_0) = -1; det(\Delta_1) = 1; det(\Delta_2) = 0; det(\Delta_3) = 0; det(\Delta_4) = 1. \tag{4.26}$$

By substituting (4.25),(4.26) into (4.24),

$$f^{-1}(x) = x^4 + x - 1 \tag{4.27}$$

From (2.15), (4.27), we get (4.3).

We need to emphasize that having $det(\Delta)$ satisfies (4.25), doesn't guarantee (4.10) to hold since the product (4.9) depends on the value of polynomial (2.16) used in (2.17) and polynomial (2.19) used in. In the following Example 4.2, we show a case when (4.25) holds but (4.10) doesn't hold and NTRU modulo p flaw is not applicable in that case.

Example 4.2 Example of failing NTRU mod $\boldsymbol{p}$ flaw when (4.25) holds but (4.10) doesn't hold.

Let according to (2.13),

$$N = 11; d = 2; p = 3; q = 40 > (6d + 1)p = 39 \tag{4.28}$$

Let us according to (2.14) and (4.25),

$$f(x) = -x^{10} - x^9 + x^7 + x^4 + x = [0,1,0,0,1,0,0,1,0,-1,-1] \tag{4.29}$$

For $N = 11$, the matrix of coefficients in (4.19) -(4.21) is as follows

$$\Delta = \begin{bmatrix} f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 \\ f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \\ f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\ f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 \\ f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 & f_4 \\ f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 & f_3 \\ f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 & f_2 \\ f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 & f_1 \\ f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} & f_0 \\ f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} \end{bmatrix} \tag{4.30}$$

To calculate the determinant of (4.30), Maple is used, Figure 4.2.

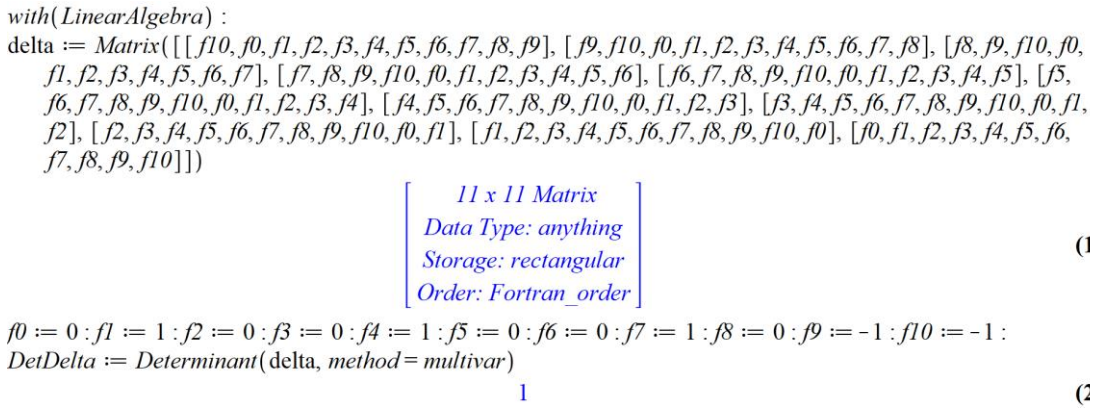

Figure 4.2: Definition of the matrix (4.30), and its determinant, in Maple 2016

Then according to (2.15),

$$F_q = 5x^{10} - 6x^9 + 7x^8 - 7x^7 + 7x^6 - 6x^5$$

$$+5x^4 - 3x^3 + x^2 + x - 3 \in R_q$$

$$F_p = 5x^{10} - 6x^9 + 7x^8 - 7x^7 + 7x^6 - 6x^5$$

$$+5x^4 - 3x^3 + x^2 + x - 3 \ mod\, p \tag{4.31}$$

Let according to (2.16), (2.18) and (2.19), $g(x)$, $r(x)$, and $m(x)$ are selected as,

$$r(x) = x^4 + x^2 - x - 1,$$

$$g(x) = x^4 + x^2 - x - 1,$$

$$m(x) = x^2 + x + 1, \tag{4.32}$$

The public key, h, according to (2.17), (4.31), (4.32) is

$$h = 15x^{10} - 14x^9 + 12x^8 - 9x^7 + 5x^6 - x^5$$

$$- 4x^4 + 8x^3 - 11x^2 + 14x - 15 \ mod\, q, \tag{4.33}$$

Ciphertext according to (2.20), (4.28), (4.32),(4.33) is:

$$e = 8x^{10} + 16x^9 + 34x^8 + 33x^7 + 23x^6 + x^5 + 30x^4$$

$$+ 16x^3 + 25x^2 + 5x + 12. \tag{4.34}$$

After obtaining ciphertext,$e(x)$ in (4.34), we try to attack (4.34) with NTRU modulo p attack steps introduced in Subsection 4.1.1.

First, we center-lift the ciphertext (4.34) w.r.t $q = 40$,

$$e = 8x^{10} + 16x^9 - 6x^8 - 7x^7 - 17x^6 + x^5 - 10x^4 + 16x^3 - 15x^2 + 5x + 12 \, mod \, q. \quad (4.35)$$

Then applying modulo p operation directly to the center-lifted ciphertext (4.35), we get the message, m' , that is not same as the plaintext message, $m(x)$, from (4.32), and, hence, the NTRU modulo p attack doesn't work in that case:

$$m' = e \, mod \, p = 2x^{10} + x^9 + 2x^7 + x^6 + x^5 + 2x^4 + x^3 + 2x \neq x^2 + x + 1. \quad (4.36)$$

Thus, Example 4.2 shows that despite (4.24) holding, condition (4.10) for NTRU modulo p applicability does not hold, and applying modulo p operation to the ciphertext (4.34) after center-lifting in (4.35), we do not get back the plaintext (4.32) in (4.36).

In Example 4.2, we used $f(x)$ (4.29) modulo $x^N - 1$, with $N = 11$ since inverses, F_q, F_p, (2.15), (4.31) have integer coefficients other than {-1,0,1} which makes holding of (4.25) and violating of (4.10) possible.

4.1.3 Estimate of the Probability of NTRU Modulo *p* Flaw

As discussed in Section 4.1.2.1, we need the determinant value, $det(\Delta)$, be equal to 1 in order to minimize coefficients (4.24). $Succ_{ModpFlaw}(N, d)$ in (4.37) defines the probability a user will choose permutation of ± 1 coefficients of (2.14) that ends up with (4.25),

$$Succ_{ModpFlaw}(N, d) = \frac{\#\, f(x) \text{ such that } |det(\Delta)| = 1}{\#\, f(x)}. \quad (4.37)$$

The number of the possible different private key, $f(x)$, the denominator of (4.37) for fixed (N, d), is found by

$$\#\, f(x) = \binom{N}{d+1}\binom{N-(d+1)}{d}$$

$$= \frac{N!}{(d+1)!\,(N-d-1)!} \cdot \frac{(N-d-1)!}{d!\,(N-2d-1)!}$$

$$= \frac{N!}{(d+1)! \cdot d! \cdot (N-2d-1)!}$$

$$= \frac{N \cdot (N-1) \cdot \ldots \cdot (N-2d)}{(d+1)! \cdot d!} = O(N^{2d+1}). \qquad (4.38)$$

According to ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009, p. 55), $1 < d < N/3$. Thus, $O(N^{2d+1})$ in (4.38) is polynomial in N, and $Succ_{ModpFlaw}(N, d)$ is not negligible (see Definition, p. 203). Howgrave et. al., (Howgrave-Graham et al., 2005), set $d = \left\lfloor \frac{N}{2} \right\rfloor$ which makes $O(N^{2d+1})$ exponential in N, and thus, $Succ_{ModpFlaw}(N, d)$ becomes negligible, Therefore, we support settings of Howgrave et. al., (Howgrave-Graham et al., 2005), to set $d = \left\lfloor \frac{N}{2} \right\rfloor$. We conducted a statistical experiment for $N = 5,\ 7,$ and 11, with $d = 1, \ldots, 5$. Table 4.1 shows number of different $f(x)$ formed such (4.25) holds for different N values and corresponding d.

Table 4.1: Numerator of (4.37) for different N values and corresponding d.

	d				
N	**1**	**2**	**3**	**4**	**5**
5	10	0	–	–	–
7	21	21	0	–	–
11	55	165	110	55	0

From Table 4.1, we can see that number of $f(x)$ such (4.25) holds increases as N growth with fixed d. It's also noticed that number of $f(x)$ such (4.25) holds equals to zero when $d = \lfloor N/2 \rfloor$, and thus mod p flaw attack fails.

$Succ_{ModpFlaw}(N, d)$ estimate roughly probability of the NTRU modulo p flaw since (4.10) most likely might happen in the cases when (4.25) holds (see Example 4.2).

4.2 Experimental Analysis of $IN-Lattice$ Attack on NTRU Private Keys

In this section, experiments were conducted (Easttom et al., 2020) to verify results in (Z. Yang et al., 2018b) and its supplementary material (Z. Yang et al., 2018a). $IN-Lattice$ Attack shown in Algorithm 2.1 has been implemented with Block-Korkine-Zolotarev(BKZ) reduction algorithm from NTL package, parameter listed in Table 4.2 are used to setup the experiments.

Table 4.2: The parameters used in our experiments.

	df	dg	dr	q
19	2	2	2	41
37	4	4	2	79
57	6	6	2	113
73	6	6	2	113
83	8	8	2	151
97	9	9	2	167
107	14	14	2	257

The value of t was recorded when a valid private key $f'$was found, and the probability $\text{prob}(f^{ls(k)} \in \mathcal{L}_I)$ was calculated using (2.50). Those results are listed in Table 4.3.

Table 4.3: The results of new attack in different ntru security levels.

N	19	37	57	73	83	97	107
t	3	5	10	13	18	27	Not found
prob	0.99999	0.99999	0.98741	0.99691	0.64895	0.12546	Not found

Contrary to results in (Z. Yang et al., 2018a), Table 4.3 shows the exponential growth of parameter t as N increases when $IN-Lattice$ attack succeeds, it means that a target vector $f^{ls(k)}$ will belong to $\mathcal{L}_I$ with low probability. Thus $IN-Lattice$ attack is infeasible for sufficiently large N (see Table 4.4). In our experiment for $N=107$, we have not got result after 6 hours of running the code, and no valid private key $f'$was found. Figure 4.3 shows an exponential growth of t as N increases.

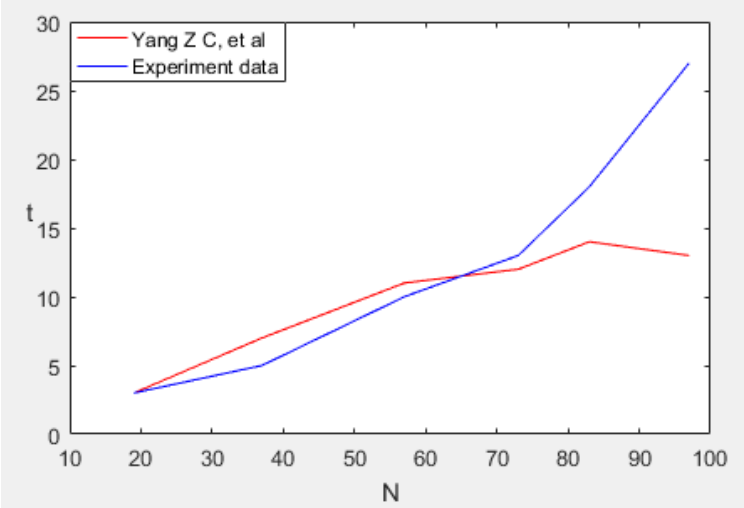

Figure 4.3: Exponential growth of t as N increases

To determine the practicality of $IN-Lattice$ attack, we used the BKZ-NTL algorithm of NTL package inside Yang's algorithm to reduce those lattices and recorded the runtime only when we found a target vector $f^{ls(k)}$ successfully. Figure 4.4 gives the results of the experiments. Time in this figure is given in seconds.

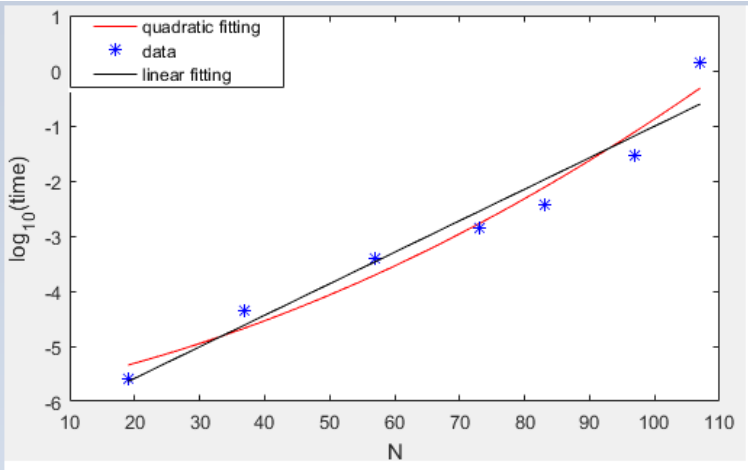

Figure 4.4: Decimal logarithm of runtime in seconds of $IN - Lattice$ Attack (blue asterisks), approximation fitting line (black), and quadratic fitting (red).

Since those experiments were run on a 2.0 GHz Core machine, the time in seconds is converted to the time in MIPS-years by first multiplying by 2.0×1024 (to account for the 2.0 GHz machine) and then dividing by 31557600 which is the number of seconds in a year (Z. Yang et al., 2018a). In this case, the experimental data were approximated by linear and quadratic fitting functions respectively as follows

$$\log_{10}(T) \approx 0.05717 \cdot N - 6.725, \tag{4.39}$$

$$\log_{10}(T) \approx 0.0002817 \cdot N^2 + 0.02158 \cdot N - 5.852 \tag{4.40}$$

Fitting curves and data are shown in Figure 4.3. The mean squared error for linear approximation is 1.063, and for quadratic approximation is 0.1077. Hence, we use the quadratic approximation for extrapolation of time for greater N values shown in Table 4.4 which shows greater time than extrapolation line in (Z. Yang et al., 2018a):

$$\log_{10}(T) \approx 0.065N - 7.3 \tag{4.41}$$

In Table 4.4, the expected time (MIPS-years) to break NTRU cryptosystem in comparison to (Z. Yang et al., 2018a) are given.

Table 4.4: Expected time (MIPS-years) to break NTRU cryptosystem in comparison to (Z. Yang et al., 2018a)

Parameter settings of NTRU	Our results	results of (Z. Yang et al., 2018a)
NTRU-167	$10^{5.61}$	$10^{3.55}$
NTRU-263	$10^{19.31}$	$10^{9.80}$
NTRU-503	$10^{76.28}$	$10^{25.4}$

From Table 4.4, we see that our expected time for $IN-Lattice$ attack is greater than shown in (Z. Yang et al., 2018a).

4.3 Summary

In this chapter, the security of NTRU is analyzed in two sections 4.1 and 4.2. In Section 4.1, a new attack on NTRU messages is proposed. The attack exploiting a flaw of NTRU allows in some cases revealing encrypted messages without the need for the private key. A numerical example of the attack is presented in Example 4.1. Probability of the attack, $O(N^{2d+1})$, is estimated in (4.38), and it is found to be polynomial in N using IEEE standard parameters. In Section 4.2, $IN-Lattice$ attack (Z. Yang et al., 2018b) is experimentally tested. Experiments are conducted using parameter settings in Table 4.2. The results of the conducted experiments in Table 4.3 and Figure 4.3 show an exponential growth of the parameter t. Thus, the attack becomes infeasible for sufficiently large N. Runtime of $IN-Lattice$ Attack is approximated by linear and quadratic fitting in (4.39) and (4.40) respectively. The quadratic fitting has less mean squared error, 0.1077, compared to, 1.063, for linear fitting. Hence, Quadratic fitting is selected for extrapolation of time for greater N. Quadratic fitting time extrapolation shown in (4.41) shows more time than extrapolation line in (Z. Yang et al., 2018a). Expected time (MIPS-years) to break NTRU cryptosystem using our quadratic fitting in comparison to linear fitting in (Z. Yang et al., 2018a) shown in Table 4.4.

NTRU, is a standardized HE scheme that is expected to be a standardized quantum homomorphic cryptosystem. Therefore, it is crucial to use the parameter setting, $d = \lfloor N/2 \rfloor$, recommended in Howgrave et. al., (Howgrave-Graham et al., 2005) to avoid NTRU mod p flaw attack.

Chapter 5

ANALYSIS OF HE1N CRYPTOSYSTEM (CLASS 4)

In this chapter, the security of HE1N (Dyer et al., 2019) from Class 4 is analyzed through three sections; Section 5.1 proves that the modulus in HE1N encryption is not working, and thus, can be omitted. In Section 5.2, a new COA against HE1N private key is designed, the complexity of the attack is found, and the success probability of the attack is estimated. In Section5.3 a new KPA against HE1N private key is designed, the complexity of the attack is found, and the success probability of the attack is estimated. Section 5.4 summarizes the results of Chapter 5.

5.1 Analysis of the Use of Modulus in HE1N Encryption

Recall from Step 4 of Algorithm 2.4 HE1N encryption:

$$c = m + sk + rp \bmod pq.$$

The modulus operation does not work since $m + sk + rp < pq$ according to Theorem 5.1 below,

Theorem 5.1: For any valid for HE1N parameters; $m \in [0, M)$; $s \in [0, k)$; $k > (n+1)^d M^d$; $r \in [1, q)$; $p > (n+1)^d (M + k^2)^d$; $q > p$, and d is the degree of polynomial homomorphically computed over n inputs, the following holds:

$$e = m + sk + rp \bmod pq = m + sk + rp, \qquad (5.1)$$

i.e., $m + sk + rp < pq$, and, hence, modulus operation does not work and can be omitted in HE1N encryption.

Proof. Let's prove that

$$m + sk + rp < pq, \tag{5.2}$$

Inequality (5.2) holds if the following holds:

$$M - 1 + k(k-1) + (q-1)p < pq, \tag{5.3}$$

or

$$M - 1 + k(k-1) + pq - p < pq, \tag{5.4}$$

or

$$k(k-1) + M - 1 < p, \tag{5.5}$$

or

$$k^2 - k + M - 1 < p. \tag{5.6}$$

Inequality (5.6) holds since by definition

$$p > (n+1)^d (M + k^2)^d > 2M + 2k^2 > k^2 - k + M - 1, \tag{5.7}$$

where $d, n \geq 1$. Thus, (5.1) is true, and, Theorem is proved.

QED

In Sections 5.2 and 5.3, new COA and KPA against HE1N private key are designed by exploiting the not-functioning modulus,

5.2 Design of Ciphertext-Only Attack (COA) Against HE1N Private Key *p*

In subsections 5.2.1-5.2.4, a new COA against HE1N private key is designed, the success probability of the attack has been estimated.

5.2.1 COA Against HE1N Private Key *p*

In this attack, the attacker is assumed to collect a set of $\mathcal{m}$ HE1N's ciphertexts. Then, the attacker applies the COA in Algorithm 5.1:

Algorithm 5.1: COA Against HE1N Private Key p

Input: a set of m HE1N's ciphertexts, $e_i = s_i k + r_i p + m_i$, since the modulus operation is not used (see Theorem 5.1)

Output: either the private key p, or the message: "key not found".

```
1.  flag = 0
2.  for i = 1 ... m do
3.    for j = 1 ... m do
4.      if i ≠ j and gcd(e_i − e_j, pq) ≠ q, then
5.        p = gcd(e_i − e_j, pq) = gcd(p(r_i − r_j), pq)
6.        flag=1;
7.        break;
8.      end if
9.    end loop// on j
10. end loop// on i
11. if flag== 1 then
12.    return p
13. else return "key not found"
```

Algorithm 5.1 succeeds to find p as the greatest common divisor of any pair, $(e_i - e_j, pq)$, $i \neq j$, if $s_i = s_j$ and $m_i = m_j$. Then, $\gcd(e_i - e_j, pq) = \gcd(k(s_i - s_j) + p(r_i - r_j) + m_i - m_j, pq) = \gcd(p(r_i - r_j), pq) = p$. Thus, the probability that the attack succeeds is the probability that among m ciphertexts can be found at least one pair (e_i, e_j), $i \neq j$ such that $s_i = s_j$ and $m_i = m_j$.

The following section discusses the problem of estimating the probability of finding a matching pair in a finite set.

5.2.2 Computational Complexity of the KPA Attack

In the worst case, Algorithm 5.1 tries step 5 for a number of runs $= m^2$, since $i, j = 1 \ldots m$. Thus, the average computational complexity of Algorithm 5.1 is,

$$\frac{m^2}{2}, \tag{5.8}$$

5.2.3 Probability of Finding a Matching Pair in a Finite Set

The birthday problem (Gorroochurn, 2012) concerns the probability of finding a matching pair in a finite set, and is defined as follows:

"*What is the probability that, in a group of m people, two of them share the same birthday?*"

Let $\mathbb{S}$ be a set of m members randomly selected from the range $[0, k)$,

$$\mathbb{S} = \{s_i\}, i = 1 \ldots m, s_i \in [0, k), \tag{5.9}$$

and let $\Pr_{no}(m)$ be the probability that among $\mathbb{S}$ no two members have the same value.

Then,

$$\Pr_{no}(m) = \Pr(s_i \neq s_j) \text{ for all } i, j = 1 \ldots m, \text{and } i \neq j, \tag{5.10}$$

$\Pr_{no}(m)$ is calculated as follows (see (Diaconis & Mosteller, 1989, sec. 7.1)):

$$\Pr_{no}(m) = \prod_{i=1}^{m} \left(1 - \frac{i}{k}\right), \text{where } 1 < m < k\,. \tag{5.11}$$

The first-order Taylor series approximation for e^{-x} can be used when x is close to zero:

$$e^{-x} \approx 1 - \frac{x}{1!} = 1 - x. \tag{5.12}$$

Then, since $\frac{i}{k} < 1$,

$$1 - \frac{i}{k} \approx e^{-i/k}, \tag{5.13}$$

and thus,

$$\Pr_{no}(m) \approx \prod_{i=1}^{m} e^{-i/k} = e^{-\frac{m(m+1)}{2k}} \approx e^{-m^2/2k}, \text{where } 1 \ll m < k\,. \tag{5.14}$$

Approximation (5.14) is greater than actual $\Pr_{no}(m)$ in (5.11) for $k \geq 2$; proof and analysis are provided in Section 5.2.4.

Let $\Pr_1(m, k)$ be the probability of having at least one pair among $\mathbb{S}$ with the same value. Then,

$$\Pr_1(m, k) + \Pr_{no}(m) = 1. \tag{5.15}$$

From (5.14) and (5.15),

$$\Pr_1(m, k) \approx 1 - e^{-m^2/2k}\,. \tag{5.16}$$

HE1N's secret parameter k is selected such that $k \approx 2^{\gamma}$. Substituting it into (5.16), one gets

$$\Pr_1(m, \gamma) \approx 1 - e^{-m^2/2^{\gamma+1}}\,. \tag{5.17}$$

Secret parameter s_i is selected from the range $[0, k)$, and the plaintext m_i is selected from the range $\left[0, \frac{k}{2}\right)$ for $n = 1$ and $d = 1$ (see (Diaconis & Mosteller, 1989, sec. 3.2.1)). Thus, from (5.16), (5.17), $\Pr_2(m, \gamma)$, the probability of having at least two pairs (s_i, s_j) and (m_i, m_j), such that $s_i = s_j$ and $m_i = m_j$ is

$$\Pr_2(m, \gamma) = \Pr_1(m, k) \cdot \Pr_1\left(m, \frac{k}{2}\right) \approx \left(1 - e^{-m^2/2k}\right)\left(1 - e^{-m^2/k}\right)$$

$$< \left(1 - e^{-m^2/k}\right)^2. \tag{5.18}$$

According to the definition, $\Pr_2(m, \gamma)$ is negligible if

$$\Pr_2(m,\gamma) < \left(1 - e^{-m^2/2^\gamma}\right)^2 < \frac{1}{\gamma^c}, \text{for any } c \geq 0 \tag{5.19}$$

for $\gamma > \gamma_c > 0$. Let us define γ_c. From (5.18), (5.19), taking square root from both sides,

$$1 - e^{-m^2/2^\gamma} < \gamma^{-\frac{c}{2}}. \tag{5.20}$$

From (5.20),

$$1 - \gamma^{-\frac{c}{2}} < e^{-m^2/2^\gamma}, \tag{5.21}$$

and, applying natural logarithm to both sides of (5.21),

$$-\ln\left(1 - \gamma^{-\frac{c}{2}}\right) > m^2/2^\gamma. \tag{5.22}$$

From (5.22), we get

$$2^{\gamma/2}\sqrt{-\ln(1 - \gamma^{-\frac{c}{2}})} > m. \tag{5.23}$$

The minimal γ for which (5.23) holds is γ_c. For example, when $m = 100, c = 10$, (5.23) holds for $\gamma \geq \gamma_{10} = 40$, and for $c = 100$, $\gamma_{100} = 455$. Thus, $\Pr_2(m,\gamma)$ is negligible.

Let us estimate the number, m, of members in a set $\mathbb{S}$ that is required to have at least two matching pairs from (5.18), is

$$\sqrt{\Pr_2(m,\gamma)} < 1 - e^{-m^2/k}, \tag{5.24}$$

and, from (5.24),

$$e^{-m^2/2^\gamma} < 1 - \sqrt{\Pr_2(m,\gamma)}. \tag{5.25}$$

By applying natural logarithm to both sides in (5.25),

$$m > 2^{\frac{\gamma}{2}}\sqrt{-\ln\left(1 - \sqrt{\Pr_2(m,\gamma)}\right)}. \tag{5.26}$$

From (5.26), the number, m, of members in a set $\mathbb{S}$ that is required to have at least two matching pairs is with $\Pr(m) = 50\%$

$$m > 2^{\frac{\gamma}{2}} \sqrt{-\ln\left(1 - \sqrt{1/2}\right)} \approx 1.11 \cdot 2^{\frac{\gamma}{2}}, \tag{5.27}$$

and with $\Pr(m) = 99.9\%$ is:

$$m > 2^{\frac{\gamma}{2}} \sqrt{-\ln\left(1 - \sqrt{0.999}\right)} \approx 2.76 \cdot 2^{\frac{\gamma}{2}}. \tag{5.28}$$

5.2.4 Analysis of $\Pr_{no}(m)$ Approximation

In the following, the proof that $\Pr_{no}(m)$ approximation, $\widetilde{Pr_{no}}(m)$, is greater than $\Pr_{no}(m)$ for $k \geq 2$.

$$\widetilde{Pr_{no}}(m) = \prod_{i=1}^{m} e^{-i/k} > \prod_{i=1}^{m} \left(1 - \frac{i}{k}\right) = Pr_{no}(m) \tag{5.29}$$

Proof: Inequality (5.29) holds if

$$e^{-i/k} > 1 - \frac{i}{k} = \frac{k-i}{k}, \tag{5.30}$$

for $k \geq 2$. Multiplying both sides of (5.30) by $e^{i/k}\ \left(\frac{k}{k-i}\right)$,

$$\frac{k}{k-i} > e^{i/k}, \tag{5.31}$$

and,

$$\left(\frac{k}{k-i}\right)^{k} > e^{i}. \tag{5.32}$$

Applying ln operation to both sides of (5.32),

$$k \ln\left(\frac{k}{k-i}\right) > i. \tag{5.33}$$

Inequality (5.33) can be written as follows:

$$k \ln k - \ln(k-i) > i. \tag{5.34}$$

By adding $\ln(k-i)$ for both sides of (5.34):

$$k \ln k > i + \ln(k-i). \tag{5.35}$$

Since $1 \leq i \leq m \leq k-1$. Then

$$k \ln k > i \ln k > i + \ln(k-i). \tag{5.36}$$

Inequality (5.36) holds for $k \geq 2$. Thus, (5.29) is proved.

QED

Figure 5.1 shows the difference between approximated, $\widetilde{Pr_{no}}(m)$, and actual probability $\Pr_{no}(m)$ for $k = 1000$ plotted by Code 5.1 From Figure 5.1, we can see that the difference is less than 0.015, and the difference becomes close to zero as m increases.

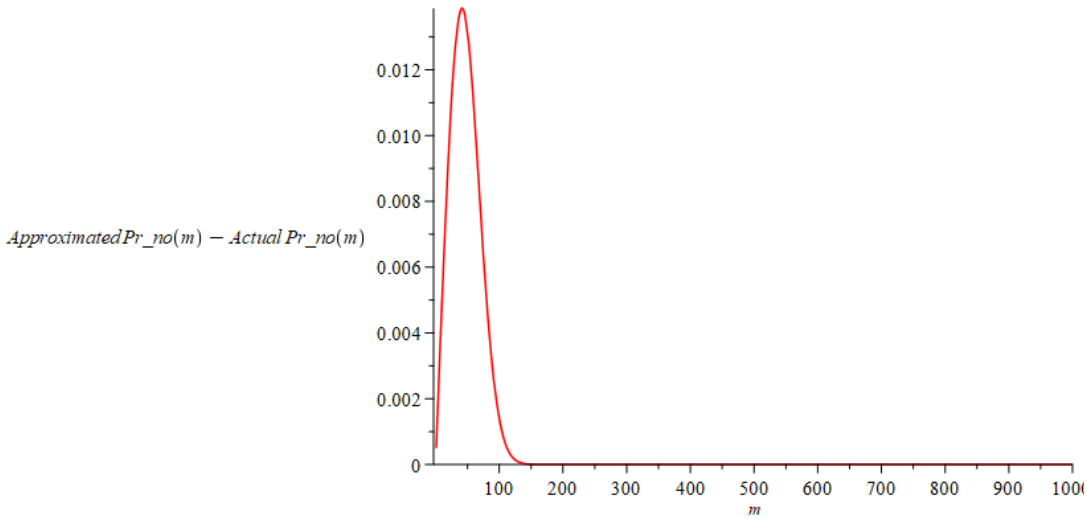

Figure 5.1: The difference between approximated and actual probability $\Pr_{no}(m)$ for $1 \leq i \leq m$, $1 \leq m \leq k$, and $k = 1000$

Code 5.1: Maple code for plotting the differences between approximated and actual $\Pr_{no}(m)$, where $1 \leq i \leq m$, $1 \leq m \leq k$, and $k = 1000$.

1. $Digits := 100;$

2. $e := \exp(1)$;
3. $k := 1000$;
4. $plot([e^{-m^2/(2*k)} - product(1 - i/k, i = 1..m)], m = 1..k, color = ["Red"])$

5.3 Design of Known Plaintext Attack Against HE1N Private Key *p*

In subsections 5.3.1-5.3.3, a new KPA against HE1N private key is designed, the complexity of the attack, the success probability of the attack has been estimated.

5.3.1 KPA Against HE1N Private Key *p*

In this attack, the attacker is assumed to collect m (ciphertext, plaintext) pairs of HE1N cryptosystem. Then the attacker applies the KPA in Algorithm 5.2:

Algorithm 5.2: KPA Against HE1N Private Key p

Input: a set of m HE1N's (ciphertext, plaintext) pairs

Output: either the private key p, or the message: "key not found".

1. flag = 0
2. $x_i = e_i - m_i = s_i k + r_i p$, where $i = 1 \ldots m$, note the modulus operation is not used (see Theorem 5.1)
3. for $j = 1 \ldots m$ do
4. for $j = 1 \ldots m$ do
5. if $i \neq j$ and $\gcd(x_i - x_j, pq) \neq 1$, then
6. set $p = \gcd(x_i - x_j, pq) = \gcd(p(r_i - r_j), pq)$, where pq is the public modulus.
7. set flag=1
8. break
9. end if
10. end loop// on *j*
11. end loop// on i
12. if flag== 1 then
13. return p

14. else return “key not found”

Algorithm 5.2 succeeds to find p as the greatest common divisor of any pair, $(x_i - x_j, pq), i \neq j$, if $s_i = s_j$. Then, $\gcd(x_i - x_j, pq) = \gcd(k(s_i - s_j) + p(r_i - r_j), pq) = \gcd(p(r_i - r_j), pq) = p$. Thus, the probability that the attack succeeds is the probability that among m (ciphertext, plaintext) pairs can be found at least one pair $(x_i, x_j), i \neq j$ such that $s_i = s_j$.

Section 5.3.2 discusses the computational complexity of Algorithm 5.2, and Section 5.4 discusses the problem of estimating the probability of finding matching pairs in a finite set.

5.3.2 Computational Complexity of The KPA Attack

In the worst case, Algorithm 5.2 tries step 5 for a number of runs $= m^2$, since $i, j = 1 \ldots m$. Thus, the average computational complexity of Algorithm 5.2 is,

$$\frac{m^2}{2}, \tag{5.37}$$

5.3.3 Probability of Finding a Matching Pair in a Finite Set

Let $\mathbb{S}$ and $\Pr_{no}(m)$ be defined in (5.9) and (5.10) respectively, $\Pr_{no}(m)$ is calculated according to (5.14).

Let $\Pr_1(m, k)$ be defined in (5.15) and calculated according to (5.17) for $k \approx 2^{\gamma}$. According to Definition, $\Pr_1(m, \gamma)$ is negligible if

$$\Pr_1(m, \gamma) < \frac{1}{\gamma^c}. \text{for any } c \geq 0 \tag{5.38}$$

for $\gamma \geq \gamma_c$. From (5.17),

$$1 - e^{-m^2/2^{\gamma+1}} < \gamma^{-c}, \tag{5.39}$$

Multiplying both sides of (5.39) by γ^c,

$$\gamma^c - 1 < e^{-m^2/2^{\gamma+1}}\gamma^c. \tag{5.40}$$

Applying natural logarithm operation to both sides of (5.40),

$$\ln(\gamma^c - 1) < \ln e^{-m^2/2^{\gamma+1}} + c\ln\gamma, \tag{5.41}$$

Thus, from (5.41)

$$\ln(\gamma^c - 1) < c\ln\gamma - \frac{m^2}{2^{\gamma+1}}, \tag{5.42}$$

and, from (5.42)

$$\frac{m^2}{2^{\gamma+1}} < \ln\left(\frac{\gamma^c}{\gamma^c - 1}\right). \tag{5.43}$$

Then, from (5.43),

$$m < \sqrt{2^{\gamma+1}\ln\left(\frac{1}{1-\gamma^{-c}}\right)}. \tag{5.44}$$

The minimal γ for which (5.44) holds is γ_c. For example, when $m = 100, c = 1$, (5.42) holds for $\gamma \geq \gamma_1 = 17$, and for $c = 5$, $\gamma_5 = 39$. Thus, $\Pr_1(m, \gamma)$ is negligible.

To estimate the required number of collected (plaintext, ciphertext) pairs to successfully attack with a certain probability, from (5.17), we get

$$1 - \Pr_1(m, \gamma) = e^{-m^2/2^{\gamma+1}}. \tag{5.45}$$

Applying natural logarithm operation to both sides of (5.45),

$$m = \sqrt{-2^{\gamma+1}\ln(1 - \Pr_1(m, \gamma))}, \tag{5.46}$$

From (5.46), the number, m, of members in a set $\mathbb{S}$ required to have at least one matching pair with $\Pr_1(m, \gamma) = 50\%$ is

$$m = \sqrt{2^{\gamma+1}\ln(2)} = \sqrt{1.39} \cdot 2^{\gamma/2} = 1.18 \cdot 2^{\gamma/2}, \quad (5.47)$$

and with $\Pr_1(m, \gamma) = 99.9\%$:

$$m = \sqrt{2^{\gamma+1}\ln(1000)} = 3.72 \cdot 2^{\gamma/2}. \quad (5.48)$$

Thus, to attack HE1N using Algorithm 5.2 with 50% success probability for $= 32$, attacker, according to (5.47), needs $m > 1.18 \cdot 2^{16} \approx 7.73 \cdot 10^4$ (plaintext, ciphertext) pairs.

The success probability of attacking HE1N by collecting $m = 100$ (plaintext/ciphertext) pairs and $\gamma = 32$ according to (5.17) is as follows:

$$\Pr_1(m, \gamma) = 1 - e^{-m^2/2k} = 1 - e^{-100^2/2^{32+1}} \approx 1.16 \cdot 10^{-6}. \quad (5.49)$$

Table 5.1 shows success probability (5.17) and computational complexity of Algorithm 5.2 (5.37), for KPA against HE1N using different m values, at $\gamma = 32$.

Table 5.1: KPA success probability and computational complexity, for different m

m	$\mathbf{Pr_1}(m, \gamma = 32)$	**Computational Complexity** $\left(\frac{m^2}{2}\right)$
100	$1.16 \cdot 10^{-6}$	$5 \cdot 10^3$
10^3	$1.16 \cdot 10^{-4}$	$5 \cdot 10^5$
10^4	0.01	$5 \cdot 10^7$
$2 \cdot 10^4$	0.05	$2 \cdot 10^8$
$5 \cdot 10^4$	0.25	$1.25 \cdot 10^9$
$7.73 \cdot 10^4$	0.50	$2.99 \cdot 10^9$
10^5	0.69	$5 \cdot 10^9$
$2.43 \cdot 10^5$	0.999	$2.95 \cdot 10^{10}$

From Table 5.1, we can see that success probability increases as the number of collected (plaintext, ciphertext) pairs increases, the success probability 50%, of having at least one matching pair can be achieved by collecting $7.73 \cdot 10^4$ pairs and the probability of 99.9% can be achieved by collecting $2.43 \cdot 10^5$ pairs.

5.4 Summary

In this chapter, HE1N security is analyzed. Section 5.1 proved that the modulus operation in HE1N encryption is effictive due to the fact that under proposed parameter settings, the value of the encryption is less than the modulus. Having ineffective modulus makes HE1N vulnerable to various attacks targeting its private key. Sections 5.2.1 and 5.3.1 new COA and KPA attacks exploiting the not functioning modulus are proposed. In sections 5.2.3, 5.2.4 the success probability of the COA attack is analyzed. The parameters γ, the length of k in bits, and m the number of collected ciphertexts can be set by eq. (5.23) so that the success probability of the COA becomes negligible. In Section 5.3.3, the success probability of KPA is estimated. The parameters γ, the length of k in bits, and m the number of collected ciphertexts can be set by eq. (5.44) so that the success probability of the KPA becomes negligible. Complexity of both attacks is found to be $O(m^2)$ by (5.37) and (5.44).

HE1N scheme is proposed as efficient homomorphic scheme to provide security for cloud computation. HE1N encryption uses an ineffective modulus operation which makes HE1N private key and encrypted message prone to various attacks such as KPA and COA attacks proposed in this chapter. In order to prevent such attacks it is important to select parameters such that the condition (5.2) is not satisfied.

Chapter 6

DEVELOPMENT OF RANDOM CONGRUENTIAL PUBLIC-KEY CRYPTOSYSTEM (RCPKC)

NTRU, and its known variants, shown in Section 2.3.4, work with degree N polynomials. The main problem NTRU faces is that it is susceptible to the lattice basis reduction attack (LBRA) using GLR algorithm (see Appendix B) for two-dimensional lattices and the LLL algorithm (see Appendix B) for higher dimensions. LBRA using LLL algorithm solves SVP with exponential in N running time revealing the secret key because the private keys are selected as polynomials with small coefficients for the decryption correctness (Hoffstein et al., 1999). To overcome the problem of susceptibility, NTRU uses large N resulting in high computational complexity (Hoffstein et al., 1998; "IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009). Therefore, NTRU variants, shown in Section 2.3.4, try minimizing NTRU computational complexity by extending the coefficients of the polynomials used or using matrices of polynomials that allow preserving the security level while decreasing the polynomial degree. The extreme case is a polynomial of zero degree, that is integers modulo $q >> 1$, as used in the congruential public key cryptosystem (CPKC), shown in Section 2.3.3, but CPKC with the NTRU encryption/encryption mechanism is insecure against LBRA by GLR (crackable in about 10 iterations) (see Example C.2 in Appendix C). Therefore, the CPKC is considered as a toy model of NTRU because "*it provides the lowest dimensional introduction to the NTRU public-key cryptosystem*" (Hoffstein et al.,

2014a, p. 374). The insecurity of CPKC stems from the choice of the private keys used as small numbers to provide decryption correctness. If CPKC could be made resistant to GLR attack, it would be the best possible choice for the NTRU modifications. Therefore, in Section 6.2, we propose a CPKC modification, random CPKC (RCPKC), working on degree $N = 0$ polynomials modulo q. The norm of a two-dimensional vector formed by its private key is greater than $\sqrt{q}$. RCPKC works as NTRU, and it is a secure version of insecure CPKC. It specifies a range from which the random numbers shall be selected and provides correct decryption for valid users and incorrect decryption for an attacker using LBRA by GLR. In Section 6.3, the security of RCPKC against various kinds of attacks is proved. In Section 6.4, RCPKC asymmetric encryption padding (RAEP), is proposed. RAEP similar to its NTRU analog, NAEP, is IND-CCA2 secure. Due to the use of big numbers instead of high degree polynomials, RCPKC is about 27 times faster in encryption and decryption than NTRU. Furthermore, RCPKC is more than three times faster than the most effective known NTRU variant, BQTRU. Compared to NTRU, RCPKC reduces energy consumption at least thirty times, which allows increasing the life-time of unattended WSNs more than thirty time. RCPKC is performance and power analysis are conducted in Section 6.5. Section 6.6 summarizes the chapter.

To modify CPKC to become resistant to GLR attack, first, in Section 6.1, a region where GLR attack fails is shown.

6.1 Region Resistant to GLR Attack on the CPKC Private Key/Message

Recall from (B.5),

$$\lambda \leq \sqrt{\gamma_2}\,\det(L)^{1/2},$$

where $\det(L) = q$ for the lattice $L(V_1, V_2)$ in (2.41). Therefore, (B.5) can be written as follows:

$$\lambda \leq \alpha\sqrt{q}, \tag{6.1}$$

where $\alpha = \sqrt{\gamma_2} \approx 1.07$. From (6.1), one gets for the relative norm,

$$\lambda' = \frac{\lambda}{\sqrt{q}}, \tag{6.2}$$

the following inequality (6.3):

$$\lambda' \leq \alpha. \tag{6.3}$$

GLR fails in attacking the CPKC private key/message when (6.3) is not satisfied for the secret vector relative norm (f, g), i.e., if:

$$\|(f, g)\|/\sqrt{q} > \alpha \tag{6.4}$$

holds, GLR fails to find the CPKC private key/message.

CPKC selects small values for the private key (f, g) in (2.29) to satisfy the decryption correctness condition (2.37). Hence, our goal is to propose in Section 6 a modification for CPKC, that is RCPKC, where (f, g) satisfies (6.4) and provides correct decryption for valid users and incorrect decryption for an attacker using GLR.

6.2 The proposed RCPKC

In this section, RCPKC, an adjustment of CPKC described in Section 2.3.3.1, so that it becomes resistant to GLR attack, is proposed.

6.2.1 RCPKC Main Ideas

The main two ideas of RCPKC are:

- Contrary to the settings (2.29) of CPKC, which uses the secret key (f, g) with a small norm not exceeding $\sqrt{q}$ so that (f, g) may be found as a shortest vector

(SV) in the lattice $E(V1, V2)$ defined by (2.41), RCPKC uses the private key (f, g) with a large norm meeting (6.4) so that it cannot be returned by LBRA using GLR as the SV, but (f, g) also meets (2.37) due to the skew in its components.

- However, as we mentioned in section 2.3.3.2, for any pair of integers, F and G, satisfying (2.42), (F, G) is likely to serve as the first two components, f, g, of the private key. That means, in spite of the large norm of (f, g), the SV=(F, G), obtained in the result of LBRA using GLR may meet decryption correctness condition (2.37) and, thus, may be used for the correct plaintext message disclosure. That is why, our proposed RCPKC before encrypting by (2.35), (contrary to CPKC using a random number from the predefined range (2.34)), defines a range for the random number selection using the SV, (F, G) (returned by GLR attack on the lattice $E(V1, V2)$ defined by (2.41)), so that decryption correctness condition (2.37) holds for (f, g) but does not hold for (F, G) which leads to the failure of LBRA using GLR on RCPKC.

Thus, RCPKC assumes that the private key owner selects a range for a random value, r (used in encryption (2.35)), based on the secret key, (f, g), and respective SV, (F, G), in the lattice, $E(V1, V2)$, defined by (2.43), guaranteeing correct decryption for a valid user and incorrect decryption for an attacker using GLR. Because of the special choice of the random value range, the proposed algorithm is called Random CPKC, RCPKC. The problem for RCPKC might happen is that the range for random numbers such as kind defined may be rather narrow and, thus, the security of RCPKC may suffer. However, as will be shown, the range is rather large and may significantly exceed the range for a secret message.

In Subsection 6.2.2, CPKC is modified to RCPKC so that it becomes immune to GLR attacks. Example F.1 in Appendix F shows GLR attack failure to disclose RCPKC encrypted message.

6.2.2 RCPKC Proposal

To meet (6.4), it is required that

$$f, r \geq \alpha \cdot \sqrt{q}. \tag{6.5}$$

The LBRA by GLR failure condition (6.4) holds if (6.5) is true since

$$\frac{\|(f,g)\|}{\sqrt{q}} = \frac{\sqrt{f^2+g^2}}{\sqrt{q}} = \frac{\sqrt{\alpha^2 \cdot q + g^2}}{\sqrt{q}} > \alpha,$$

for $g > 0$. Condition (6.5), in RCPKC, substitutes for the conditions (2.29), (2.34) on f, r, in CPKC. The message, m, and the private key, g, instead of (2.33), (2.29), used in CPKC, are redefined in RCPKC as follows:

$$2^{mgLen} > g \geq 2^{mgLen-1} > m \geq 0 \tag{6.6}$$

where $mgLen$ represents the length of m and g in bits.

For RCPKC, correctness decryption condition (2.37) shall hold, that is true (see (6.11)) when f, r values in addition to (6.5) meet (6.7):

$$\frac{q}{2 \cdot 2^{mgLen}} > f, r. \tag{6.7}$$

since $q = 2^{qLen}$. Then, (6.5), (6.7) can be rewritten:

$$2^{qLen-mgLen-1} > f, r \geq \alpha \cdot 2^{qLen/2}. \tag{6.8}$$

To have a non-empty range for f, r, of the width at least $\alpha \cdot 2^{qLen/2}$, from (6.8), the following condition is obtained:

$$\frac{2^{qLen/2}}{2 \cdot \alpha} > 2^{mgLen+1}. \tag{6.9}$$

By defining $\beta = \log_2 1/(2 \cdot \alpha) \approx -1.103$, (6.9) shows that

$$2^{\beta} \cdot 2^{qLen/2} > 2^{mgLen+1},$$

$$qLen + 2 \cdot \beta > 2 \cdot (mgLen + 1),$$

$$qLen > 2 \cdot (mgLen + 1 - \beta). \quad (6.10)$$

Let's show that the decryption correctness condition (2.37) holds when (6.6), (6.8), and (6.10) hold:

$$r \cdot g + f \cdot m < 2^{qLen-mgLen-1} \cdot 2^{mgLen} + 2^{qLen-mgLen-1} \cdot 2^{mgLen-1}$$

$$< 2^{qLen-1} + 2^{qLen-1} = 2^{qLen} = q. \quad (6.11)$$

Thus, for RCPKC, the norm (f, g) meets (6.4) and the decryption correctness condition (6.11) holds. We need additionally that decryption correctness condition (6.11) to be violated for (F, G), that is the SV obtained in the result of GLR attack on the lattice $E(V1, V2)$ defined by (2.41). Hence, it cannot be used as a private key for the plaintext message correct decryption.

All vectors (F_i, G_i) obtained in the course of GLR reduction that have norms

$$\|(F_i, G_i)\| < \mu\|(f, g)\|, i = 1, \dots, N, \quad (6.12)$$

must be listed, where N is the number of (F, G) pairs satisfying (6.12), μ is a threshold, e.g., $\mu = 10$, and then it must be checked that

$$(\forall i = 1, \dots, N)\big((F_i, G_i) \neq (f, g)\big). \quad (6.13)$$

If (6.13) is violated, i.e., one of the vectors in the list is our vector (f, g), then another (f, g) is used.

Inequality (6.8) defines a range for r so that f, g, r, m meet (2.37). Now, constraint on r is defined as follows:

$$\frac{q}{g} - f \geq rmax \geq r \geq rmin \geq \left(q + g \cdot \max_{i=1,\dots,N} |F_i|\right) \Big/ \min_{i=1,\dots} |G_i|, \tag{6.14}$$

such that F_j, G_j, r, m violate (2.37) for any $j = 1, \dots, N$. We require also that

$$h \cdot rmin > q. \tag{6.15}$$

Using (6.14) and (6.6), it is noticed that actually decryption correctness condition (2.37) for any $j = 1, \dots, N$, is violated:

$$\left|G_j \cdot r + F_j \cdot m\right| \geq \left|G_j \cdot r\right| - \left|F_j \cdot m\right| \geq \left|G_j\right| \cdot \frac{q + g \cdot \max\limits_{i=1,\dots,N} |F_i|}{\min\limits_{i=1,\dots,N} |G_i|} - \left|F_j \cdot m\right|$$

$$\geq q + g \cdot \max_{i=1,\dots,N} |F_i| - \left|F_j \cdot m\right| > q. \tag{6.16}$$

From (6.6), (6.14), it is also perceived that the decryption correctness condition (2.37) holds for the original (f, g):

$$g \cdot rmax + f \cdot m \leq g(q/g - f) + f \cdot m = q - f(g + m) < q. \tag{6.17}$$

Thus, inequality (6.8) is used for f, but for r from (6.14) and (6.8):

$$rmax > r \geq \max\left(\alpha \cdot 2^{qLen/2}, rmin\right). \tag{6.18}$$

For RCPKC security, the range defined by (6.18) shall be rather large, such as, e.g., $\max(\alpha \cdot 2^{qLen/2}, rmin)$, hence, it is desirable having:

$$rmax \geq 2 \cdot \max\left(\alpha \cdot 2^{qLen/2}, rmin\right). \tag{6.19}$$

To provide CCA indistinguishability (see Section Definition G.28), it is required to have

$$\gcd(g, q) > 1. \tag{6.20}$$

Thus, RCPKC proposal follows.

RCPKC proposal:

The private key components, (f, g), meet (2.30), (2.31), (6.6), (6.8), and (6.20), where $qLen$, $mgLen$ meet (6.10). The public key component, h, is defined by (2.32). Message, m, meets (6.6), and random integer, r, is selected from the range defined in (6.14), (6.15) and (6.18). Encryption and decryption follow (2.35), (2.36) and (2.39), respectively. The decryption correctness condition (2.37) is proven for RCPKC in (6.17).

Example F.1 shown in Appendix F is an example of RCPKC encryption and decryption, and GLR failure to find RCPKC private key/message. RCPKC is also resistant to various attacks, as shown in the security analysis presented in the next section.

6.3 Security Analysis

In this section, attacks on NTRU are considered: brute force (on the key and message), and meet-in-the-middle (MITM) in Subsection 6.3.1, lattice basis reduction in Subsection 6.3.2, hybrid lattice basis reduction, and MITM (Howgrave-Graham, 2007) in Subsection 6.3.3, multiple transmission (MTA) (Hoffstein et al., 1998) in Subsection 6.3.5, and also, the most recent, chosen-ciphertext (Gama & Nguyen, 2007; Hoffstein et al., 2017; Howgrave-Graham, Nguyen, et al., 2003; Jaulmes & Joux, 2000), in Subsection 6.3.5), and we try applying them to RCPKC. Herein, the NTRU parameters used, EES401EP1 ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009), of the security level, $k = 112$ bits:

$$N = 401, p = 3, q = 2048, df_1 = df_2 = 8,$$
$$df_3 = 6, dg = 133, dr_1 = dr_2 = 8, dr_3 = 6 \quad (6.21)$$

To meet the same security level, the RCPKC settings satisfying (6.10) are:

$$qLen = 473, mgLen = 255, \tag{6.22}$$

The key space cardinality (defined in Subsection 6.3.1 for the parameters (6.21) and (6.22)) is greater than or equal to $2^{2 \cdot k}$ for $k = 112$ to avoid the MITM attack explained in Subsection 6.3.1.

6.3.1 Brute Force and MITM Attacks

An attacker can recover the NTRU private key by trying all possible values of g and testing whether $f \cdot h \bmod q$ has small coefficients (the product corresponds to g according to (2.32)). On the other hand, an attacker can try all possible values of g and test whether $h^{-1} \cdot g \bmod q$ (corresponding to f by virtue of (2.32)) has small coefficients. Equations (6.23) and (6.24) show the search space cardinalities for g and f for the security level, $k = 112$ (taking into account the MITM attack explained later in this section). The search space cardinality for f is computed as follows (see (Hoffstein et al., 2017) (Section 7)):

$$\begin{aligned} C_{NTRU}(f,k) &= \binom{N}{df_1}\binom{N-df_1}{df_1}\binom{N}{df_2}\binom{N-df_2}{df_2}\binom{N}{df_3}\binom{N-df_3}{df_3} \\ &= \binom{401}{8}\binom{393}{8}\binom{401}{8}\binom{393}{8}\binom{401}{6}\binom{395}{6} \\ &= 1.16 \times 10^{90} \geq 2^{2 \cdot k} = 2^{224}. \end{aligned} \tag{6.23}$$

Similarly, for g:

$$\begin{aligned} C_{NTRU}(g,k) &= \binom{N}{df_1}\binom{N-df_1}{df_1} = \binom{401}{133}\binom{268}{133} \\ &= 4.34 \times 10^{188} \geq 2^{2 \cdot k} 2^{224}. \end{aligned} \tag{6.24}$$

it is perceived the search space cardinality for f is less than that for g, so the best strategy for an attacker is to search for f values.

An attacker can reduce the search space cardinality from 2^k to $2^{k/2}$ (Howgrave-Graham, Silverman, & Whyte, 2003) using MITM by splitting the private key f (which is a polynomial of degree $N-1$) into two polynomials, $f = f_1 + f_2$, where f_1 is a polynomial of degree at most $N/2 - 1$ and polynomial f_2 contains terms of degree between $N/2$ and $N-1$, and then trying matches: $f_1 \cdot h \bmod q = (g - f_2 \cdot h) \bmod q$. Hence, to meet the $k = 112$ security level, the NTRU parameters must be chosen to meet the $k = 224$ security level, as it is already made in (6.21). For RCPKC, the secret value, g, is selected from the interval $[2^{mgLen-1}, 2^{mgLen})$ (see (6.6)); hence, the search space cardinality for g to meet the $2 \cdot k$-bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(g,k) = 2^{mgLen-1} \geq 2^{2 \cdot k} \tag{6.25}$$

The secret value, f, is selected from the interval $[\alpha \cdot 2^{qLen/2}, 2^{qLen-mgLen-1})$ (see (6.8)); hence, the search space cardinality for f to meet the $2 \cdot k$-bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(f,k) = 2^{qLen-mgLen-1} - \alpha \cdot 2^{qLen/2} \geq 2^{2 \cdot k} \tag{6.26}$$

For the parameters (6.22), $C_{RCPKC}(g,k) = 2^{224}$, while $C_{RCPKC}(f,k) \approx 2^{247}$. In order to provide the security level for $k = 112$, the parameters (6.22) are chosen to meet the twice greater security level of $2 \cdot k = 224$ to counter the MITM attack, considered below, which reduces the brute force attack effort by the square root. Since $C_{RCPKC}(g,k) < C_{RCPKC}(f,k)$, the best strategy for an attacker is to search for g values. Similar to NTRU, the MITM attack can be applied to the RCPKC private key component, g. Since $mgLen$ is the bit length of g, then $g = g_1 + 2^{(mgLen-1)/2} \cdot g_2$, and then, g_1 and g_2 , each of a bit length equal to $(mgLen - 1)/2$, can be enumerated with the resulting search space cardinality $\mathcal{O}(2^{(mgLen-1)/2})$ trying to find matching

$$(f \cdot h - g_1)\ mod\ q = 2^{(mgLen-1)/2} \cdot g_2\ mod\ q.$$

Thus, the RCPKC parameters (6.22) provide the security level $k = 112$ against the brute force attack with MITM. Now, let us consider the brute force attack on the message.

An attacker can compromise an NTRU message by trying all possible values of r and testing whether $e - r \cdot h\ mod\ q$ has small coefficients. Similarly, the attacker can compromise the RCPKC message by trying all possible values of r and testing if $e - r \cdot h\ mod\ q\ \in\ [0, 2^{mgLen-1})$ by virtue of (6.6).

The RCPKC message search space is defined by the interval $[0, 2^{mgLen-1})$ (see (6.6)); hence, the search space cardinality f or m to meet the $2 \cdot k$-bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(m, k) = 2^{mgLen-1} \geq 2^{2 \cdot k}, \tag{6.27}$$

while the search space of r is defined by (6.14), (6.18), and (6.19). Hence, the search space cardinality for r to meet the $2 \cdot k$-bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(r, k) = rmax - \max\left(\alpha \cdot 2^{qLen/2}, rmin\right) \geq 2^{2 \cdot k}. \tag{6.28}$$

Table 6.1: Width of the range for the r value for different security levels.

#	**RCPKC Parameter**	$\mathbf{2 \times k}$		
		224	336	448
1	$mgLen$	225	337	450
2	$qLen$	473	743	909
3	$f = 2^{qLen-mgLe-1} - 1$	2.26×10^{74}	8.26×10^{121}	7.44×10^{137}
4	g	$2^{mgLen} - 1 = 5.39 \times 10^{67}$	$2^{mgLen} - 5 = 2.79 \times 10^{101}$	$2^{mgLen} - 11 = 2.90 \times 10^{135}$
5	$rmax$	2.26×10^{74}	8.26×10^{121}	7.44×10^{137}
6	$\max\left(\alpha 2^{qLen/2}, rmin\right)$	7.41×10^{72}	1.62×10^{119}	1.10×10^{137}

7	$c_{RCPKC}(r,k)$	2.1×10^{74}	8.24×10^{121}	6.34×10^{137}

Table 6.1 shows the $mgLen$ and $qLen$ values to meet different $2\cdot k$-bit security levels' condition (6.28) (see Rows 1 and 2) and the width of the range for r (Row 7) with f and g specified in Rows 3 and 4, respectively. It proves that the method can be practically used.

6.3.2 Lattice Basis Reduction Attacks

The NTRU lattice basis, L_h^{NTRU}, associated with public key h defined in (2.17) is

$$L_h^{NTRU}=\left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right),$$

where $h_0,\ldots,h_{N-1}$ are coefficients of the polynomial h. For convenience, matrix L_h^{NTRU} is abbreviated as

$$L_h^{NTRU}=\begin{pmatrix} I & h \\ 0 & qI \end{pmatrix}.$$

The NTRU private key recovery problem can be formulated as the SVP in $2N$-dimensional lattice, L_h^{NTRU}. If a polynomial, b, of degree $N-1$ with integer coefficients satisfying:

$$f\cdot h+q\cdot b=g$$

exists, then:

$$(f,b)=L_h^{NTRU}=(f,g).$$

Therefore, the vector (f, g) is in the lattice L_h^{NTRU}. Vector (f, g) or its rotation (rotation of a polynomial, f, by i steps, is $x^i \cdot f \in Rq$ for an integer i) can be found if it is the shortest vector in L_h^{NTRU}. The lattice reduction algorithm LLL (see Appendix B) finds the shortest vector in L_h^{NTRU} in time exponential in N. According to (Hoffstein et al., 1999), LLL takes 1.05×1031 MIPS (million instructions per second)-years to find the shortest vector or its rotation for $N = 400$ (as in (6.21)) that most likely is the NTRU private key part, (f, g).

Contrary to NTRU, RCPKC is resistant to LBRA since the GLR attack fails for it (see Subsection 6.2.2). LBRA is one of the most used and effective techniques in attacking an NTRU private key (e.g., it is used in the hybrid lattice attack, the most efficient on practical NTRU parameters (Kirchner & Fouque, 2017); see Subsection 6.3.4), but it does not apply to RCPKC.

6.3.3 A Hybrid Lattice Basis Reduction and MITM Attack

The attack (Howgrave-Graham, 2007) on the NTRU secret key combines the LBRA and MITM strategies. The hybrid attack, first, splits the original lattice of order $2N$, $N > 1$, into three subparts, only one of which is further reduced, whereas the vectors from the other parts are just enumerated, thus combining the concepts of the LBRA and MITM attacks. The hybrid attack is not applicable to RCPKC since:

- The RCPKC lattice is two-dimensional and cannot be split into three subparts;
- RCPKC uses a large norm secret (f, g) vector (see (6.6) and (6.8)) that cannot be found by LBRA looking for an SV, and the SV cannot be used for correct decryption (see (6.16))

6.3.4 Multiple Transmission Attack

MTA reveals a large part of an NTRU message by sending n times the same message, m, using the same public key, h, but different random values, r^i. For NTRU encryption (2.35)

$$e^i r^i \cdot h + m \; mod \; q$$

for $i = 1,2, \dots, n$. An adversary computes:

$$\left(e^i - e^1\right) \cdot h^{-1} \; mod \; q,$$

There by recovering $r^i - r^1 \; mod \; q$, $i = 1, \dots, n$, and from these relations, many coefficients of r^1 may be revealed. Knowledge of r^1 allows disclosing the message, m. RCPKC is not susceptible to MTA because no special structure is assumed for r^1 contrary to the case of NTRU.

6.3.5 Chosen Ciphertext Attack

Three chosen-ciphertext attacks (CCA) on NTRU are known. The first key recovering CCA described in (Jaulmes & Joux, 2000) uses a ciphertext of a special shape, which can be countered by message padding (Hoffstein et al., 2017). Standardized parameters (Hoffstein et al., 2017) allow decryption failure, i.e., a ciphertext could fail to be decrypted correctly by NTRU. In (Howgrave-Graham, Nguyen, et al., 2003), a CCA was presented where an attacker collects a large number of decryption failures; see the NTRU correction decryption condition (2.29) in Subsection 2.3.1. Another CCA was presented in (Gama & Nguyen, 2007), which is more efficient than (Howgrave-Graham, Nguyen, et al., 2003), but still depends on decryption failures. RCPKC works on non-structured integers, and the parameters, set in Subsection6.2.2, guarantee correct decryption. Thus, neither of the CCAs described above apply to RCPKC.

6.4 RCPKC Asymmetric Encryption Padding and its IND-CCA2 Security

In this section, we prove the security of the RCPKC one-way function based on the discussions of the security of the NTRU one-way function in (Howgrave-Graham et al., 2005), define RCPKC asymmetric encryption padding (RAEP), and prove its IND-CCA2 security as a particular case of NAEP (see Appendix G). According to Section 6.2.2, RCPKC defines the following four sets:

- $\mathcal{D}_f = \left[\alpha \cdot 2^{qLen/2}, 2^{qLen-mgLen-1}\right)$: private key space, an interval from which a private key, f, is selected;
- $\mathcal{D}_g = [2^{mgLen-1}, 2^{mgLen})$: private key space, an interval from which a private key, g, is selected;
- $\mathcal{D}_m = [0,2^{mgLen-1})$: RCPKC plaintext space, an interval from which a plaintext, m, is selected;
- $\mathcal{D}_r = \left[\max\left(\alpha \cdot 2^{\frac{qLen}{2}}, rmin\right), rmin\right]$: RCPKC random value space.

The RCPKC encryption primitive is specified by the parameter set, $\mathcal{P} = \left(q, \mathcal{D}_f, \mathcal{D}_g, \mathcal{D}_m, \mathcal{D}_r\right)$. The one-way function underlying RCPKC is:

$$F_h: \mathcal{D}_m \times \mathcal{D}_r \to \mathbb{Z}_q,$$

$$F_h(m, r) = r \cdot h + m \bmod q.$$

Definition 6.1. RCPKC-OW problem: For a parameter set, $\mathcal{P}$, we denote by $Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P})$ the success probability of a PPT adversary, $\mathcal{A}$, for finding a pre-image of F_h,

$$Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P}) = \Pr\left(\begin{matrix}(m', r') \leftarrow \mathcal{A}(e, h) \\ such\ that (\exists r' \in \mathcal{D}_r)(F_h(m', r') = e)\end{matrix}\right).$$

Assumption 6.1: RCPKC-OW assumption: For every PPT adversary, $\mathcal{A}$, solving the RCPKC-OW problem, there exists a negligible function, $v_A(k)$, such that for sufficiently large k, we have:

$$Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P}) \leq v_A(k).$$

An adversary $\mathcal{A}_1$ can compromise (m, r) by picking $r' \in \mathcal{D}_r$, substituting it in $(e - r' \cdot h)\ mod\ q$, and checking, if the result is in $\mathcal{D}_m$. Thus, $Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P})$ is:

$$Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P}) = \frac{2^{mgLen}}{2^{qLen}}.$$

Since $qLen > mgLen$ by definition (6.10), $Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P})$ decreases exponentially in $qLen$, and Assumption holds. Similarly, the attacker can try the following methods with an exponentially decreasing success probability:

1. The adversary, $\mathcal{A}_2$, chooses randomly a pair $(r' \in \mathcal{D}_r\ , m' \in \mathcal{D}_m)$ and checks if $r' \cdot h + m'\ mod\ q = e$.
2. The adversary, $\mathcal{A}_3$, picks $f' \in \mathcal{D}_f$, substitutes it in $f' \cdot h\ mod\ q$, and checks whether the result is in $\mathcal{D}_g$.
3. The adversary, $\mathcal{A}_4$, chooses randomly a pair $\left(f' \in \mathcal{D}_f, g' \in \mathcal{D}_g\right)$, if possible, calculates h', decrypts e to (r', m'), and checks if $r' \cdot h' + m'\ mod\ q = e$.
4. Furthermore, the adversary can apply the GLR attack to get $(f\ , g)$. However, by construction, RCPKC is immune to that attack, and hence, the success probability is zero. Therefore, the Assumption is true for all the above attacks.

RCPKC encryption (2.35) differs from NTRU encryption (2.20) just by setting $N = p = 1$. The conclusion of (Howgrave-Graham, Silverman, Singer, et al., 2003) on NAEP IND-CCA2 security is also true for asymmetric encryption padding, RAEP. However, NAEP cannot be used as-is for $N = p = 1$ because it

utilizes specific true polynomial functions center() and compress(). Since the decryption correctness condition (2.37) holds for RCPKC due to the parameter choice, the center() function is not used in RCPKC and RAEP. The function compress() as in NAEP shall map its input, $p \cdot r \cdot h$, to a binary string, bs, of the padded message size. In NAEP, it is done in two steps: $s = compress(p \cdot r \cdot h \, mod q)$; $bs = H(s)$. In RAEP, both transforms are done by one hash function, $H: \mathbb{Z}_q \rightarrow \{0,1\}^{mgLen}$. Algorithm 6.1 and Algorithm 6.2 show RAEP encryption and decryption, respectively.

Algorithm 6.1: RAEP encryption algorithm

Input: $N = mgLen = \theta(k)$ is the length of the RCPKC encrypted message; $N > l = \theta(k)$ is the padding size; $G: \{0,1\}^{N-l} \times \{0,1\}^{l} \rightarrow \mathcal{D}_r$ and $H: \mathbb{Z}_q \rightarrow \{0,1\}^{N}$ are hash functions; $\mathcal{D}_r$ is defined by (6.13), (6.17) and (6.18); $m \in \{0,1\}^{N-l}$ is the input plaintext message.

Output: $e \in \mathbb{Z}_q$ is the ciphertext.

1. Pick $\mu \overset{\$}{\leftarrow} \{0,1\}^{l}$
2. Let $\rho = H(m, \mu)$, $r = genr(\rho)$, $s = r \cdot h \, mod \, q$, and $\omega = (m||\mu) \oplus H(s)$// $genr(\quad)$ is a function generating correct r according to (6.14), (6.15), (6.18), and (6.19).
3. Let $e = F_h(m, r)$ // according to (2.35)

Algorithm 6.2: RAEP decryption algorithm

Input: $N = \theta(k)$ is the length of the RCPKC encrypted message; $N > l = \theta(k)$ is the padding size; $G: \{0,1\}^{N-l} \times \{0,1\}^{l} \rightarrow \mathcal{D}_r$ and $H: \{0,1\}^{N} \rightarrow \{0,1\}^{N}$ are hash functions; $\mathcal{D}_r$ is defined by the space for r; $e \in \mathbb{Z}_q$ id the ciphertext.

Output: $m \in \{0,1\}^{N-l}$ is the decrypted plaintext message if decrypted correctly, and Reject, otherwise.

1. $a = f \cdot e \, mod \, q$ // according to (2.36)

2. $\omega = F_g \cdot a \; mod g$ // according to (2.39).
3. $s = e - \omega \; mod \; q$
4. $m||\mu = \omega \oplus H(s); r = genr\big(G(m||\mu)\big)$
5. if $r \cdot h = s \; mod \; q$
6. output m
7. else
8. output Reject

Evaluations of performance and power consumption are provided in Section6.5.

6.5 RCPKC Performance and Power Consumption Evaluation

6.5.1 RCPKC Performance Evaluation

Experiments were conducted using the NTRU code (*GitHub - NTRUOpenSourceProject/Ntru-Crypto: Open Source NTRU Public Key Cryptography and Reference Code*, n.d.) and RCPKC implementation in the C99 language similar to (*GitHub - NTRUOpenSourceProject/Ntru-Crypto: Open Source NTRU Public Key Cryptography and Reference Code*, n.d.) with the NTL library (*NTL: A Library for Doing Number Theory*, n.d.) on a PC equipped with 1.6 GHz Intel Core i5-8250U, 8 GB RAM, and Windows 10 (see the main part of RCPKC source code is available in Appendix J, and the full code is available in (*Anasnet/RCPKC-Project: Open Source RCPKC Cryptosystem Reference Code*, n.d.)). Both the NTRU code (*GitHub - NTRUOpenSourceProject/Ntru-Crypto: Open Source NTRU Public Key Cryptography and Reference Code*, n.d.) and the proposed RCPKC were implemented in Visual Studio 2017. The NTRU parameters (6.21) and the RCPKC parameters (6.22) were used. The CPU encryption and decryption time of RCPKC and NTRU was measured for 10^3, 10^4, and 10^5 runs. In each run, a distinct 128-bit message was encrypted/decrypted with both cryptosystems. The NTL function

RandomLen() was used to pseudo-randomly generate the messages. RandomLen() was seeded with the output of the function clock(). The generated messages were stored in a separate file and used to test RCPKC and NTRU. The CPU time was measured via QueryPerformanceCounter() with ns accuracy.

Table 6.2 shows the sample mean, $\bar{x}$, standard deviation, σ, and confidence interval with the confidence level $C \in \{0.95, 0.99, .0.999\}$ for the number of runs $n \in \{103, 104, 105\}$, respectively for RCPKC and NTRU. The confidence interval, $[l, u]$, is calculated using (Moore, 2006, p. 350):

$$[l, u] = \left[\bar{x} - z^* \frac{\sigma}{\sqrt{n}}, \bar{x} + z^* \frac{\sigma}{\sqrt{n}}\right], \tag{6.29}$$

where $\bar{x} = (\sum_{i=1}^{n} x_i)/n$, $\sigma = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}/(n-1)$, x_i, and n are the sample mean, sample standard deviation, the value of the run, and number of runs, respectively; z^* is the critical value required for the specific confidence level; see Table C in (Moore, 2006, p. 687). For example, in Table 6.2 for RCPKC encryption with $C = 95\%$, $n = 103$, $\bar{x} = 6.19 \times 10^{-6}$, $\sigma = 3.966 \times 10^{-6}$, $z^* = 1.960$, the confidence interval is calculated as follows:

$$[l, u] = \left(6.190 \times 10^{-6} - \frac{1.960\left(3.966 \times 10^{-6}\right)}{\sqrt{1000}}, 6.190 \times 10^{-6} + \frac{1.960\left(3.966 \times 10^{-6}\right)}{\sqrt{1000}}\right) =$$

$(6.112 \times 10^{-6}, 6.267 \times 10^{-6})$.

Table 6.2: RCPKC and NTRU CPU encryption/decryption time sample mean, standard deviation, and confidence interval for different runs

Algorithm	**Measured Value**	**Run number**		
		10^3	10^4	10^5
	C	0.95	0.99	0.999
	z^*	1.960	2.576	3.291
	Sample Mean, $\bar{x}$	6.190×10^{-6}	5.492×10^{-6}	4.708×10^{-6}

RCPKC (Encryption)	Sample Standard Deviation, σ	3.966×10^{-6}	2.076×10^{-6}	2.923×10^{-6}
	Confidence Interval $[l, u]$	$(6.112 \times 10^{-6}, 6.267 \times 10^{-6})$	$(5.475 \times 10^{-6}, 5.508 \times 10^{-6})$	$(4.677 \times 10^{-6}, 4.738 \times 10^{-6})$
NTRU (Encryption)	Sample Mean, $\bar{x}$	1.444×10^{-4}	1.964×10^{-4}	1.440×10^{-4}
	Sample Standard Deviation, σ	6.878×10^{-5}	1.123×10^{-5}	6.437×10^{-5}
	Confidence Interval $[l, u]$	$(1.430 \times 10^{-4}, 1.457 \times 10^{-4})$	$(1.430 \times 10^{-4}, 1.973 \times 10^{-4})$	$(1.430 \times 10^{-4}, 1.447 \times 10^{-4})$
RCPKC (Decryption)	Sample Mean, $\bar{x}$	9.506×10^{-6}	8.812×10^{-6}	7.493×10^{-6}
	Sample Standard Deviation, σ	2.781×10^{-6}	2.370×10^{-6}	2.795×10^{-6}
	Confidence Interval $[l, u]$	$(9.451 \times 10^{-6}, 9.560 \times 10^{-6})$	$(8.792 \times 10^{-6}, 8.831 \times 10^{-6})$	$(7.464 \times 10^{-6}, 7.522 \times 10^{-6})$
NTRU (Decryption)	Sample Mean, $\bar{x}$	2.079×10^{-4}	2.826×10^{-4}	2.088×10^{-4}
	Sample Standard Deviation, σ	9.700×10^{-5}	1.594×10^{-4}	8.633×10^{-5}
	Confidence Interval $[l, u]$	$(2.060 \times 10^{-4}, 2.098 \times 10^{-4})$	$(2.813 \times 10^{-4}, 2.839 \times 10^{-4})$	$(2.079 \times 10^{-4}, 2.097 \times 10^{-4})$

Figure 6.1 shows the NTRU/RCPKC encryption and decryption average CPU time ratio for $10^3, 10^4$, and 10^5 runs. From Figure 6.1, it is observed that RCPKC is 27.08 ± 3.75 times faster than NTRU in encryption and 26.9 ± 5.09 times faster in decryption, respectively.

Table 6.3 compares NTRU versus RCPKC and several NTRU variants presented in Subsection 2.3.4. It is observed that RCPKC is faster than the fastest most recently published NTRU variant, BQTRU, more than four times in encryption.

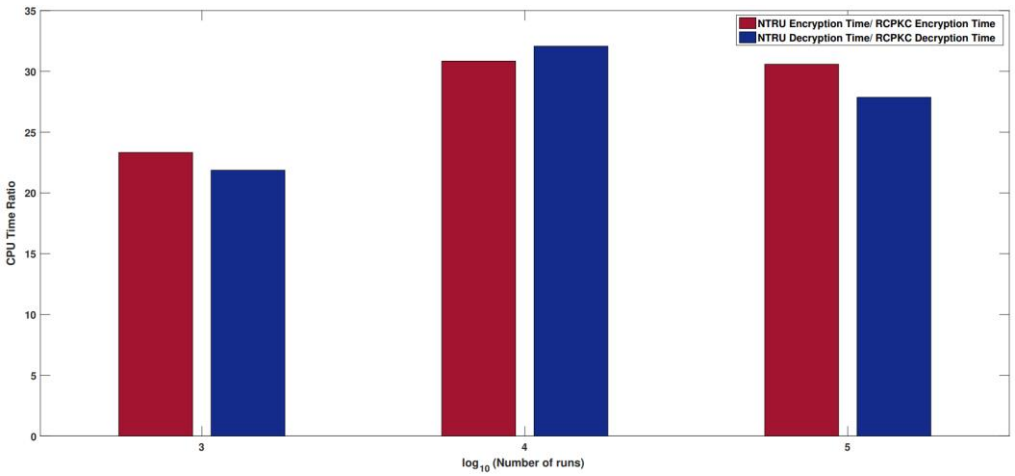

Figure 6.1: NTRU/RCPKC encryption and decryption average CPU time ratio for 10^3, 10^4, and 10^5 runs.

Table 6.3: Ratios of encryption and decryption times of NTRU and the variants $A \in \{RCPKC, BQTRU, MaTRU, ETRU\}$.

Algorithm, *A*	**Encryption Time Ratio,** $= \frac{T_{NTRU}^{ENC}}{T_A^{ENC}}$	**Decryption Time Ratio,** $= \frac{T_{NTRU}^{DEC}}{T_A^{DEC}}$
Proposed RCPKC	27	27
BQTRU (Bagheri et al., 2018)	7	No Data
MaTRU (Coglianese & Goi, 2005)	2.5	2.5
ETRU (Jarvis & Nevins, 2015)	1.45	1.75

From Table 6.3, we can see that RCPKC is faster than the fastest most recently published NTRU variant, BQTRU, more than four times in encryption.

6.5.2 RCPKC Power Consumption Evaluation

In this section, RCPKC's power consumption is compared to NTRU in two cases: applying both algorithms using the same or different frequencies.

Same frequencies: Let the RCPKC and NTRU execution time be T_{RCPKC} and T_{NTRU}, respectively. Then, from equation (H.1) in Appendix H, the consumed energy by NTRU and RCPKC E_{NTRU} and E_{RCPKC} is:

$$E_{NTRU} = P \cdot T_{NTRU} \quad \text{and} \quad E_{RCPKC} = P \cdot T_{RCPKC}. \tag{6.30}$$

Since T_{NTRU} is greater than T_{RCPKC} by more than 27 times, then from (6.30):

$$\frac{E_{NTRU}}{E_{RCPKC}} = \frac{T_{NTRU}}{T_{RCPKC}} \geq 27. \tag{6.31}$$

From (6.31), RCPKC consumes twenty-seven times less energy than NTRU using the same frequency.

Different frequencies: Since RCPKC is 27 times faster than NTRU, the former takes approximately the same run time on 27 times lower clock frequency CPU than that of the latter. Dynamic and leakage power consumption, calculated for frequencies from (Texas Instruments Incorporated, 2018, p. 19) according to (H.5) in Appendix H, are shown in Table 6.4.

Table 6.4: Microcontroller MSP430FR5969 dynamic and leakage power consumption, P_{dyn} and P_{leak}, for different frequencies and active supply voltages.

	2.2 V		**3 V**	
Frequency (MHz)	$\boldsymbol{P_{dyn}}$ **(μW)**	$\boldsymbol{P_{leak}}$ **(nW)**	$\boldsymbol{P_{dyn}}$ **(μW)**	$\boldsymbol{P_{leak}}$ **(nW)**
1	48.4	44	90	60
16	774.4		1440	

It follows from Table 6.4 that $P_{leak} \ll P_{dyn}$, and it can be neglected. From Table 6.4, it follows that reducing the clock frequency from 16 to 1 MHz leads to a 16 times power consumption reduction from 1440 to 90 μW. Note that MSP430FR5969, at a lower frequency, operates at a lower voltage: operating on a 1 MHz frequency at 2.2V results in 48.4 μW of dynamic power consumption. Hence, the total power reduction

is $\frac{1440}{48.4} \approx 30$ times. Therefore, RCPKC, compared to NTRU, is better applicable to WSNs with power-constrained devices.

6.6 Summary

In this chapter, RCPKC is proposed, a secure and effective congruential, modulo q, public-key cryptosystem using big numbers. It uses the same encryption/decryption mechanism as NTRU does, but works with numbers. Contrary to NTRU, RCPKC is resistant to LBRA because its private key components, f, and g, are chosen w.r.t $\sqrt{q}$ to form a two-component vector with the norm exceeding Minkowski's boundary (B.4), (6.1)-(6.3) for the shortest vector in a two-dimensional lattice and meeting (6.4). Hence, LBRA by the GLR algorithm returning the shortest vector in a two-dimensional lattice fails at finding the large norm private key vector, (f, g).

Despite the big numbers, f and r, meeting (6.8) used in RCPKC, it guarantees that the decryption correctness condition (2.37) holds (see (6.11)) due to the use of Conditions (6.6), (6.8), (6.10), (6.14) and (6.18) instead of Conditions (2.29), (2.33), and (2.34), used in the original insecure CPKC (see Section 2.3.3.1) considered in (Hoffstein et al., 2014a). It was found that the insecurity of the original CPKC stems from the use of Conditions (2.29), (2.33), and (2.34), defining smaller than $\sqrt{q}$ numbers f, g, m, r meeting Minkowski's boundary (B.4) and the decryption correctness condition (2.37). RCPKC is resistant to the LBRA by GLR attack due to the special choice of the range for the random value, r, used in the encryption (2.35) that guarantees correctness condition (2.37) violation for the short vectors returned by GLR, but holding for the original private key, (f, g). Section 6.3 shows also that the security of RCPKC with respect to other known attacks on NTRU is not less than that of NTRU, which allows

us to conclude that RCPKC is more secure than NTRU. Section 6.4 proves the IND-CCA2 security of RCPKC asymmetric encryption padding (RAEP).

RCPKC uses numbers, i.e., minimal possible, degree zero, polynomials, which makes it about 27 times more effective in encryption and decryption than NTRU and more than three times more effective in encryption w.r.t the fastest most recently published NTRU variant, BQTRU (Bagheri et al., 2018), as the experiments show (see Table 6.3). Compared to NTRU, RCPKC reduces the energy consumption at least 27 times, which allows increasing the life-time of unattended WSNs by more than 27 times.

The efficiency and security of proposed RCPKC allows it to be used to provide security of various applications running on battery dependent devices such as smartphones, and devices with limited computational power such as medical devices and sensors.

Chapter 7

DEVELOPMENT OF FULLY HOMOMORPHIC CRYPTOSYSTEM WITHOUT NOISE CONTROL MECHANISM (RLWE-CSCM)

HE schemes based on LWE such as (Brakerski et al., 2014, 2013; Brakerski & Vaikuntanathan, 2011a, 2011b) from Class 6 need NCM and CSCM. In this chapter, RLWE-CSCM, the first HE scheme supports two arithmetic operations and doesn't need NCM by construction. i.e., the growth of noise doesn't lead to decryption failure by construction. In Section 7.1, RLWE-CSCM is proposed advancing a previously proposed RLWE-NCM-CSCMin (Brakerski & Vaikuntanathan, 2011b). In Section 7.2, homomorphism of RLWE-CSCM w.r.t addition and multiplication are proved. In Section 7.3, the security of RLWE-CSCM against several attacks is presented. Section 7.4 summarizes Chapter 7.

7.1 Proposed RLWE-CSCM

RLWE-CSCM is a public-key cryptosystem that works in the ring $R = \mathbb{Z}[x]/(x^d + 1)$. of RLWE-CSCM parameter definition follows

7.1.1 Parameter Setup

Select integers p and q such that

$$q = \alpha \cdot p. \tag{7.1}$$

Let rings $R = \mathbb{Z}[x]/(x^d + 1)$, and $R_n = \mathbb{Z}_n[x]/(x^d + 1)$, where parameters $\alpha > 1$, $p > 1$, $d > 1$, $n \in \mathbb{Z}^+$.

7.1.2 Key Generation

Let A, s, and e be selected from R_q, where A is not invertible meets

$$\deg(gcd(A, x^d + 1)) \neq 0, \tag{7.2}$$

where $\deg(\mathbf{a})$ is the degree of a polynomial **a**. The public key, $pk \in R_q^2$, is computed as follows:

$$pk = \begin{pmatrix} pk_1 \\ pk_2 \end{pmatrix} = \begin{pmatrix} A \cdot s + p \cdot e \\ -A \end{pmatrix}. \tag{7.3}$$

Secret key, $sk \in R_q^2$, is:

$$sk = \begin{pmatrix} sk_1 \\ sk_2 \end{pmatrix} = \begin{pmatrix} 1 \\ s \end{pmatrix}. \tag{7.4}$$

7.1.3 Encryption

Let r be a random polynomial selected from R_q. Encryption of the message, $m \in R_p$, is computed as follows

$$c \leftarrow Enc_{pk,r}(m) = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r + m \\ pk_2 \cdot r \end{pmatrix} \in R_q^2. \tag{7.5}$$

7.1.4 Anti Ciphertext Only Attacks Condition

Two ciphertext-only attacks (COA) are shown in subsections 7.3.17.3.2 and 7.3.2. To counter these attacks, the message owner selects a random r in (7.5), such that

$$\deg(c_2) \leq \deg(m), \tag{7.6}$$

and,

$$\exists\, i \in \mathbb{Z}_d : r_i \bmod p > 0, \tag{7.7}$$

where r_i is the i-th coefficient of r.

7.1.5 Decryption

The process of decryption is:

$$m' = Dec_{sk}(c) = \left[[\langle c, sk \rangle]_q\right]_p, \tag{7.8}$$

where $\langle a, b\rangle$ is the inner product of vectors a and b, and $[a]_q = a \; mod \; q$.

7.1.6 Proof of Decryption Correctness

Theorem 7.1 RLWE-CSCM decryption (7.8) of c encrypted by (7.5) is correct, i.e., $m' = m$, if (7.1) holds. If (7.1) is not true, decryption fails with probability $1 - (1/p^d)$.

Part 1: If (7.1) holds and ciphertext c is encrypted by (7.5), then decryption by (7.8) results in $m' = m$.

Proof of Part 1: According to (7.8), decryption is performed in two steps:

Step 1: According to (7.3)-(7.8),

$$\begin{aligned} [\langle c, sk\rangle]_q &= c_1 \cdot sk_1 + c_2 \cdot sk_2 \; mod \; q, \\ &= (pk_1 \cdot r + m) \cdot 1 + (pk_2 \cdot r) \cdot s \; mod \; q \\ &= (A \cdot s + p \cdot e) \cdot r + m + (-A \cdot r \cdot s) \; mod \; q \\ &= p \cdot e \cdot r + m - k \cdot q, \end{aligned} \tag{7.9}$$

where $k \in R$. Recalling (7.1), one gets from (7.9),

$$[\langle c, sk\rangle]_q = m + p \cdot (e \cdot r - \alpha \cdot k). \tag{7.10}$$

Step 2: Modulo p operation applied to (7.10) vanishes the contributor with p and leaves the message m:

$$m' = \left[[\langle c, sk\rangle]_q\right]_p = m + p \cdot (e \cdot r - \alpha \cdot k) \; mod \; p = m. \tag{7.11}$$

Part 1 is proved.

Part 2: If (7.3)-(7.5) are satisfied, but (7.1) doesn't hold, then $m' \neq m$ with the probability $1 - (1/p^d)$.

Proof of Part 2: Let $q = \alpha \cdot p + t, t > 0,\ t \in \mathbb{Z}_p$. Then, from (7.9),

$$[\langle c, sk \rangle]_q = p \cdot e \cdot r + m - k \cdot q \tag{7.12}$$

$$= m + p \cdot e \cdot r - k \cdot \alpha \cdot p - k \cdot t$$

$$m - k \cdot t + p \cdot (e \cdot r - k \cdot \alpha), \tag{7.13}$$

where $k \in R$. In Step 2 of decryption, the message m' retrieved by applying modulo p operation to the last expression in (7.13) is

$$m' = m - k \cdot t + p \cdot (e \cdot r - k \cdot \alpha)\ mod\ p = m - k \cdot t\ mod\ p. \tag{7.14}$$

From (7.14),

$$m' - m = -k \cdot t\ mod\ p.$$

Therefore, $m' \neq m$ for $k \neq 0\ mod\ p, k \in R$. Assuming all coefficients' values are equally likely, the probability of having all coefficients $k_i = 0\ mod\ p, i \in Z_d$, is $(\alpha/q)^d = (\alpha/(\alpha \cdot p))^d = 1/p^d$. Thus, the probability of having at least one coefficient $k_i \neq 0\ mod\ p,\ i \in Z_d$, i.e., probability of $m' \neq m$ is $1 - (1/p^d)$. Thus, Part 2 is proved.

QED.

Example I.1 in Appendix I is an example of correct encryption/decryption when (7.1) holds, and Example I.2 in Appendix I is an example of incorrect decryption when (7.1) does not hold.

Homomorphism of RLWE-CSCM w.r.t addition and multiplication is proved in Section 7.4

7.2 Homomorphism of RLWE-CSCM with Respect to Addition and Multiplication

7.2.1 Homomorphic Addition

First, homomorphism of single addition operation is proved in Subsection 7.2.1.1. Then, homomorphism for any number of additions is proved in Subsection 7.2.1.2

7.2.1.1 Single Homomorphic Addition

Theorem 7.2: Let $c^{(i)}, i \in \{1,2\}$ is a ciphertext encrypting the plaintext messages $m^{(i)}$ using random $r^{(i)}$ according to (7.5),

$$c^{(i)} = \begin{pmatrix} c_1^{(i)} \\ c_2^{(i)} \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r^{(i)} + m^{(i)} \\ pk_2 \cdot r^{(i)} \end{pmatrix} \in R_q^2, i = 1,2. \tag{7.15}$$

Then, RLWE-CSCM decryption (7.8) of $C = c^{(1)} + c^{(2)}$ is $m' = m^{(1)} + m^{(2)}$, i.e., RLWE-CSCM is homomorphic w.r.t single addition.

Proof: Let C be the sum of the ciphertexts $c^{(1)}$ and $c^{(2)}$, then,

$$C = c^{(1)} + c^{(2)} = \begin{pmatrix} c_1^{(1)} + c_1^{(2)} \\ c_2^{(1)} + c_2^{(2)} \end{pmatrix}$$

$$= \begin{pmatrix} \left(r^{(1)} + r^{(2)}\right)pk_1 + \left(m^{(1)} + m^{(2)}\right) \\ \left(r^{(1)} + r^{(2)}\right)pk_2 \end{pmatrix} \tag{7.16}$$

Decrypting of C is performed by (7.8) as follows:

$$[\langle C, sk \rangle]_q$$

$$= \left(\left(r^{(1)} + r^{(2)}\right)pk_1 + \left(m^{(1)} + m^{(2)}\right)\right)1 + \left(\left(r^{(1)} + r^{(2)}\right)pk_2\right)s \; mod \; q$$

$$= \left(r^{(1)} + r^{(2)}\right)(A \cdot s + p \cdot e) + \left(m^{(1)} + m^{(2)}\right) + \left(r^{(1)} + r^{(2)}\right)(-A \cdot s) \; mod \; q$$

$$= \left(r^{(1)} + r^{(2)}\right)p \cdot e + \left(m^{(1)} + m^{(2)}\right) \; mod \; q$$

$$= \left(m^{(1)} + m^{(2)}\right) + \left(r^{(1)} + r^{(2)}\right)p \cdot e - k \cdot q,$$

where $k \in R$. Recalling (7.1), one gets

$$[\langle C, sk\rangle]_q = \left(m^{(1)} + m^{(2)}\right) + \left(\left(r^{(1)} + r^{(2)}\right) \cdot e - \alpha \cdot k \right) p. \tag{7.17}$$

In Step 2 of decryption, modulo p operation applied to (7.21) vanishes the contributor with p and leaves the message $\left(m^{(1)} + m^{(2)}\right)$:

$$m' = \left[\langle C, sk\rangle_q\right]_p$$
$$= \left(m^{(1)} + m^{(2)}\right) + \left(\left(r^{(1)} + r^{(2)}\right)e - \alpha \cdot k \right) \cdot p \bmod p = m^{(1)} + m^{(2)}. \tag{7.18}$$

QED.

Example I.3 in Appendix I, is an example of the homomorphic addition of two terms. Homomorphism for any number of additions follows.

7.2.1.2 Homomorphism for Any Number of Additions

Theorem 7.3: If $c^{(i)}$, is a ciphertext encrypting the plaintext message $m^{(i)}$ using random $r^{(i)}$ according to (7.5),

$$c^{(i)} = \begin{pmatrix} c_1^{(i)} \\ c_2^{(i)} \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r^{(i)} + m^{(i)} \\ pk_2 \cdot r^{(i)} \end{pmatrix} \in R_q^2, i = 1..N. \tag{7.19}$$

Then, RLWE-CSCM decryption (7.8) of $C = c^{(1)} + \cdots + c^{(N)}$ is $m' = m^{(1)} + \cdots + m^{(N)}$, i.e., RLWE-CSCM is homomorphic w.r.t any number N of additions.

Proof: Let $c^{(i)}$ be a ciphertext defined in (7.19), and let

$$C = c^{(1)} + \cdots + c^{(N)} = \begin{pmatrix} c_1^{(1)} + \cdots + c_1^{(N)} \\ c_2^{(1)} + \cdots + c_2^{(N)} \end{pmatrix}. \tag{7.20}$$

Decrypting of C is performed by (7.8) as follows:

$$[\langle C, sk\rangle]_q = \left(\left(r^{(1)} + \cdots + r^{(N)}\right)pk_1 + \left(m^{(1)} + \cdots + m^{(N)}\right)\right)1$$
$$+ \left(\left(r^{(1)} + \cdots + r^{(N)}\right)pk_2\right)s \bmod q$$

$$= \left(r^{(1)} + \cdots + r^{(N)}\right)(A \cdot s + p \cdot e) + \left(m^{(1)} + \cdots + m^{(N)}\right)$$

$$+ \left(r^{(1)} + \cdots + r^{(N)}\right)(-A \cdot s)\ mod\ q$$

$$= \left(r^{(1)} + \cdots + r^{(N)}\right) \cdot p \cdot e + \left(m^{(1)} + \cdots + m^{(N)}\right)\ mod\ q$$

$$= \left(m^{(1)} + \cdots + m^{(N)}\right) + \left(r^{(1)} + \cdots + r^{(N)}\right) \cdot p \cdot e - k \cdot q,$$

where $k \in R$. Recalling (7.1), one gets

$$[\langle C, sk\rangle]_q = \left(m^{(1)} + \cdots + m^{(N)}\right) + \left(\left(r^{(1)} + \cdots + r^{(2)}\right)e - \alpha \cdot k\right) \cdot p. \qquad (7.21)$$

In Step 2 of decryption, the modulo p operation applied to (7.21) vanishes the contributor with integer p and leaves the message $\left(m^{(1)} + \cdots + m^{(N)}\right)$:

$$m' = \left[\langle C, sk\rangle_q\right]_p$$

$$= \left(m^{(1)} + \cdots + m^{(N)}\right) + \left(\left(r^{(1)} + \cdots + r^{(N)}\right)e - \alpha \cdot k\right) p\ mod\ p$$

$$= m^{(1)} + \cdots + m^{(N)}. \qquad (7.22)$$

Thus, homomorphism for any number of additions is proved. Corollary 7.1 follows immediately from Theorem 7.3:

Corollary 7.1: If C is encryption of $N \cdot m$, then decryption of C by (7.8) results in $N \cdot m, N \in Z$.

Corollary 7.1 means that RLWE-CSCM is also homomorphic w.r.t multiplication by any integer.

Example I.4 in Appendix I, is an example of homomorphic for $1000c^{(1)} + 2000c^{(2)}$.

7.2.2 Homomorphic Multiplication of Ciphertexts

In this section, the homomorphism of RLWE-CSCM w.r.t multiplication is explained. In Subsection 7.2.2.1, homomorphism for single multiplication is proved, and the

growth of ciphertext size due to homomorphic multiplication is explained. In Subsections 7.2.2.2-7.2.2.4, two CSCMs are presented; ciphertext recryption; ciphertext re-linearization.

7.2.2.1 Homomorphism for Single Multiplication

Definition 7.1. Let $c^{(1)}$ and $c^{(2)}$ be the ciphertexts of the plaintexts $m^{(1)}, m^{(2)}$ obtained by (7.5). Product of the ciphertexts, $M\left(c^{(1)}, c^{(2)}\right)$, is defined in (7.23)

$$M\left(c^{(1)}, c^{(2)}\right) = \begin{pmatrix} c_1^{(1)} c_1^{(2)} \\ c_1^{(1)} c_2^{(2)} + c_2^{(1)} c_1^{(2)} \\ c_2^{(1)} c_2^{(2)} \end{pmatrix} \tag{7.23}$$

Theorem 7.4: If $c^{(i)}$, is a ciphertext encrypting the plaintext messages $m^{(i)}$ using random $r^{(i)}$ according to (7.5), $i \in \{1,2\}$, then, RLWE-CSCM decryption (7.26), using $sk3$ (7.25), of $M(c^{(1)}, c^{(2)})$ is $m' = m^{(1)} \cdot m^{(2)}$, i.e., RLWE-CSCM is homomorphic with respect to a single multiplication.

$$m' = Dec_{sk3}\big(M(c1, c2)\big) = \big[[\langle M(c1, c2), sk3\rangle]_q\big]_p. \tag{7.24}$$

Note that Dec_{sk3} is a scalar product of 3-component vectors.

$$sk3 = \begin{pmatrix} 1 \\ s \\ s^2 \end{pmatrix}. \tag{7.25}$$

Proof: Let $c^{(i)}$ be ciphertext encrypting message $m^{(i)}$, $i = 1,2$, and sk is the secret key (7.4). Then,

$$\langle c^{(1)}, sk\rangle = c_1^{(1)} + c_2^{(1)} \cdot s,$$

$$\langle c^{(2)}, sk\rangle = c_1^{(2)} + c_2^{(2)} \cdot s,$$

and

$$\langle c^{(1)}, sk\rangle \cdot \langle c^{(2)}, sk\rangle = (c_1^{(1)} + c_2^{(1)} \cdot s)(c_1^{(2)} + c_2^{(2)} \cdot s)$$

$$= c_1^{(1)}c_1^{(2)} + \left(c_1^{(1)}c_2^{(2)} + c_2^{(1)}c_1^{(2)}\right)s + c_2^{(1)}c_2^{(2)}s^2 = \langle M(c1, c2), sk3\rangle, \quad (7.26)$$

where $M(c1, c2)$ in (7.23) and $sk3$ in (7.25). Decryption $C^{Mul(1,2)}$ is performed using the secret key $sk3$ according to (7.24).

Steps of computing $Dec_{sk3}(M(c1, c2))$:

Step 1: Calculate $[\langle M(c1, c2), sk3\rangle]_q$ by (7.26),

$$[\langle M(c1, c2), sk3\rangle]_q = \left[\langle c^{(1)}, sk\rangle \cdot \langle c^{(2)}, sk\rangle\right]_q = \langle M(c1, c2), sk3\rangle$$

$$= \left(m^{(1)} + p \cdot \left(e \cdot r^{(1)} - \alpha \cdot k\right)\right) \cdot \left(m^{(2)} + p \cdot \left(e \cdot r^{(2)} - \alpha \cdot k\right)\right) \quad (7.27)$$

Step 2: By (7.27),

$[\langle M(c1, c2), sk3\rangle]_q \; mod \; p$

$$= \left(m^{(1)} + p\left(e \cdot r^{(1)} - \alpha \cdot k_1\right)\right)\left(m^{(2)} + p\left(e \cdot r^{(2)} - \alpha \cdot k_2\right)\right) mod \; p$$

$$= m^{(1)} \cdot m^{(2)} \quad (7.28)$$

QED.

Example I.5 in Appendix I is an example of a single homomorphic multiplication. As shown in (7.23) and (7.25), the dimension of the ciphertext of the product of two plaintexts and the respective decryption key is three, contrary to the original RLWE-CSCM encryption (7.5) with the dimension of two. Thus, the complexity of the decryption process (7.26) and required storage increases with the increase of the number of multipliers, and extending the multiplication to many terms will have more and more increasing complexity. To avoid the dimension rise, two CSCM's are proposed; ciphertext recryption; ciphertext linearization are proposed in Subsections 7.2.2.2, 7.2.2.4 respectively.

7.2.2.2 Homomorphic Multiplication Using Recryption

Recryption allows returning result of homomorphic multiplication of two ciphertexts to the form of standard single plaintext encryption (7.5) but additionally encrypted by multiplication with a secret recryption constant. Dimension of the ciphertext product, $(c^{(1)}, c^{(2)}) \in R_q^3$, in (7.23), three, is greater than the dimension, two, of RLWE-CSCM encryption, $c \in R_q^2$, in (7.5). Hence, the complexity of decryption process will increase for each multiplication which makes it infeasible to extend the multiplication to many terms. To solve this problem, a ciphertext recryption algorithm is proposed. It takes $M(c^{(1)}, c^{(2)}) \in R_q^3$, a recryption key, k_{rec}, which is the secret key multiplied by a recryption constant, $K \in \mathbb{Z}_p$, and the public key as inputs, and outputs a RLWE-CSCM ciphertext, $C\_K\_m1\_2 \in R_q^2$, that encrypts by (7.5) the product of two plaintexts multiplied by K. The principle steps of the ciphertext recryption follow:

1. Decrypt $M\left(c^{(1)}, c^{(2)}\right) \in R_q^3$, in (7.23) according to (7.24) using recryption key, k_{rec}

$$k_{rec} = K \cdot sk3 = \begin{pmatrix} K \\ Ks \\ Ks^2 \end{pmatrix} \in R_q^3, \tag{7.29}$$

where $K \in \mathbb{Z}_p$ is a secret recryption constant that is defined by the owner of the secret key (7.25). The decryption process results in

$$km_{1\_2} \leftarrow Dec_{k_{rec}}\left(M\left(c^{(1)}, c^{(2)}\right)\right) = K \cdot m^{(1)} \cdot m^{(2)} \tag{7.30}$$

(Proof of (7.30) is provided next page).

And thus, the secret key, $sk3$, in (7.29) and the product $m^{(1)} \cdot m^{(2)}$ in (7.30) are encrypted by K. Recryption constant, K, can be removed using multiplicative inverse, K^{-1} existing if K meets,

$$\gcd(K, p) = 1. \tag{7.31}$$

2. Next, km_{1_2} is encrypted using pk, according to (7.5), obtaining $C\_K\_m1\_2 \in R_q^2$. Therefore, by applying the ciphertext recryption process after each multiplication, the ciphertext will be defined by (7.5), and hence, there is no increase in the decryption complexity or the memory needed to store the ciphertext and respective keys.

Proof of (7.30): Let k_{rec} defined in (7.29) and $M\left(c^{(1)}, c^{(2)}\right)$ in (7.23), then $Dec_{k_{rec}}\left(M\left(c^{(1)}, c^{(2)}\right)\right)$ is computed as follows:

Step 1: Calculate $[\langle M(c1, c2), k_{rec}\rangle]_q$ by (7.26),

$$[\langle M(c1, c2), k_{rec}\rangle]_q$$
$$= c_1^{(1)} \cdot c_1^{(2)} \cdot K + \left(c_1^{(1)} \cdot c_2^{(2)} + c_2^{(1)} \cdot c_1^{(2)}\right) \cdot K \cdot s + c_2^{(1)} \cdot c_2^{(2)} \cdot K \cdot s^2$$
$$= \left[\langle c^{(1)}, sk\rangle \cdot \langle c^{(2)}, K \cdot sk\rangle\right]_q \quad (7.32)$$

Step 2: By (7.32),

$$\left[[\langle M(c1, c2), k_{rec}\rangle]_q\right]_p$$
$$= \left(m^{(1)} + p\left(e \cdot r^{(1)} - \alpha \cdot k\right)\right)\left(K \cdot m^{(2)} + K\right.$$
$$\left. \cdot\, p\left(e \cdot r^{(2)} - \alpha \cdot k\right)\right) mod\ p$$
$$= K \cdot m^{(1)} \cdot m^{(2)}. \quad (7.33)$$

QED.

Recryption key, k_{rec} (7.29), reveals secret recryption constant, K, as the first component. Thus, the secret key, s, can be found from the second component as well. Compromises s leads to discover secret key, sk, and thus all encrypted messages can be compromised by an unauthorized party. This issue happens due to the structure of

sk, where the first component is constant equals to 1. To overcome this issue, a new structure of sk is proposed with no constants.

Proposed changes to sk and the consequent modifications are as follows:

A. parameters $\beta \in \mathbb{Z}_q, \gamma \in \mathbb{Z}_p$ are selected such that,

$$\beta - \gamma = \omega \cdot p \bmod q, \omega \in \mathbb{Z}_q, \tag{7.34}$$

and,

$$\gcd(\beta, s) > 1\,, \gcd(\gamma, s) > 1, \gcd(\beta, K) > 1, \gcd(\gamma, p) = 1. \tag{7.35}$$

B. Instead of (7.4), secret key, sk, is defined as follows

$$sk = \begin{pmatrix} sk_1 \\ sk_2 \end{pmatrix} = \begin{pmatrix} \beta \\ \gamma \cdot s \end{pmatrix}. \tag{7.36}$$

C. Instead of the message, m, its product with the recryption constant, K, is encrypted using (7.5),

$$c \leftarrow Enc_{pk,r}(mK) = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r + mK \\ pk_2 \cdot r \end{pmatrix} \in R_q^2, \tag{7.37}$$

$$mK = m \cdot K \in R_p. \tag{7.38}$$

The decryption process of ciphertext, c (7.37), encrypting mK (7.38) is performed using sk (7.36) as follows:

$$mK' \leftarrow Dec_{sk}(c) = \left[[\langle c, sk \rangle]_q \cdot \gamma^{-1}\right]_p \tag{7.39}$$

D. Recryption key, $sk3$ (7.40), is used instead of (7.25),

$$sk3 = \begin{pmatrix} \beta^2 \\ \gamma \cdot \beta \cdot s \\ (\gamma \cdot s)^2 \end{pmatrix}. \tag{7.40}$$

Derivation of $sk3$ (7.40) is as follows:

$$\langle c^{(1)}, sk \rangle \cdot \langle c^{(2)}, sk \rangle = \left(c_1^{(1)} \cdot \beta + c_2^{(1)} \cdot \gamma \cdot s\right) \cdot \left(c_1^{(2)} \cdot \beta + c_2^{(2)} \cdot \gamma \cdot s\right)$$

$$= c_1^{(1)} \cdot c_1^{(2)} \cdot \beta^2 + \left(c_1^{(1)} \cdot c_2^{(2)} + c_2^{(1)} \cdot c_1^{(2)}\right) \cdot \beta \cdot \gamma \cdot s + c_2^{(1)} \cdot c_2^{(2)} \cdot (\gamma \cdot s)^2$$

$$= \langle M(c1, c2), sk3 \rangle, \tag{7.41}$$

$M(c1, c2)$ is defined in (7.23).

E. Recryption key, k_{rec} (7.29), is computed using $sk3$ (7.40), as follows

$$k_{rec} = K \cdot sk3 = \begin{pmatrix} K \cdot \beta^2 \\ K \cdot \gamma \cdot \beta \cdot s \\ K \cdot (\gamma \cdot s)^2 \end{pmatrix} \in R_q^3, \tag{7.42}$$

Thus, in (7.42), the secret recryption constant, K, is encrypted using secret parameters β, γ, and s, contrary to (7.29).

Decryption of $M(c1, c2)$ (7.23) using k_{rec} (7.42) is performed as follows,

$$mK' \leftarrow Dec_{k_{rec}}\big(M(c1, c2)\big)$$

$$= \left[[\langle M(c1, c2), k_{rec} \rangle]_q (K\beta^2)^{-1}\right]_p = mK^{(1)} \cdot mK^{(2)}$$

$$= K^2 m^{(1)} m^{(2)}, \tag{7.43}$$

where $mK^{(i)} = K * m^i$ is the padded message of $m^{(i)}$ according to (7.37).

Proof of (7.43): Let k_{rec} defined in (7.42) and $M\big(c^{(1)}, c^{(2)}\big)$ in (7.23), then $Dec_{k_{rec}}\left(M\big(c^{(1)}, c^{(2)}\big)\right) = \left[[\langle M(c1, c2), k_{rec} \rangle]_q (K\beta^2)^{-1}\right]_p$ is computed as follows:

Step 1: Calculate $[\langle M(c1, c2), k_{rec} \rangle]_q$ by (7.26),

$$[\langle M(c1, c2), k_{rec} \rangle]_q$$

$$= c_1^{(1)} c_1^{(2)} \cdot K \cdot \beta^2 + \left(c_1^{(1)} c_2^{(2)} + c_2^{(1)} c_1^{(2)}\right) \cdot K \cdot \gamma \cdot \beta \cdot s + c_2^{(1)} c_2^{(2)}$$

$$\cdot K \cdot (\gamma \cdot s)^2 = \left[\langle c^{(1)}, sk \rangle \cdot \langle c^{(2)}, K \cdot sk \rangle\right]_q$$

$$= \left(\beta \cdot mK^{(1)} + p \cdot \left(e \cdot r^{(1)} - \alpha \cdot k_1\right)\right)$$

$$\cdot \left(K \cdot \beta \cdot mK^{(2)} + p \cdot \left(K \cdot \beta \cdot e \cdot r^{(2)} - \alpha \cdot k_2\right)\right). \tag{7.44}$$

Step 2. By (7.44),

$$\left[[\langle M(c1, c2), k_{rec}\rangle]_q (K\beta^2)^{-1}\right]_p$$

$$= mK^{(1)} \cdot mK^{(2)} \bmod p = K^2 \cdot m^{(1)} \cdot m^{(2)}. \quad (7.45)$$

QED.

The process of getting a ciphertext, $C\_K\_m1\_2$, of (7.43) using the product of two ciphertexts and recryption is defined in Algorithm 7.1(MultRecrypt). Algorithm 7.2(DecRecrypt) describes the process of decrypting $C\_K\_m1\_2$, obtaining the product of two plaintexts $m^{(1)}$ and $m^{(2)}$.

Algorithm 7.1 MultRecrypt: Algorithm of getting RLWE-CSCM ciphertext, $C\_K\_m1\_2$ (7.5) of (7.43) using product of two RLWE-CSCM ciphertexts and Recryption.

Input: $c^{(1)}$, $c^{(2)}$: ciphertexts of the messages $mK^{(1)} = K \cdot m^{(1)}$, $mK^{(2)} = K \cdot m^{(2)}$, respectively, using (7.5); pk, public key; k_{rec}, recryption key in (7.42)

Output: $C\_K\_m1\_2 \in R_q^2$, mk' encrypted by (7.5) using pk

1. Set $Mc1\_2 \leftarrow M\left(c^{(1)}, c^{(2)}\right)$, where $M\left(c^{(1)}, c^{(2)}\right)$ is the product of ciphertexts $c^{(1)}$ and $c^{(2)}$ defined in (7.23).;
2. Decrypt $Mc1\_2$ using k_{rec} according to (7.24), obtaining mK' (7.43)
3. Set $C\_K\_m1\_2 \leftarrow Enc_{(pk,r)}(mK') = \begin{pmatrix} pk_1 \cdot r + mK' \\ pk_2 \cdot r \end{pmatrix} \in R_q^2$
4. **Return** $C\_K\_m1\_2$

Proving that the proposed last variant has no problems as the first one

---

Algorithm 7.2 DecRecrypt: Getting product of the plaintexts from the output of Algorithm 7.1MultRecrypt.

Input:	$C\_K\_m1\_2$, recrypted ciphertext output by Algorithm 1; $sk3$, in (7.40), K, recryption constant meeting (7.31).
Output:	$m' \in R_p$, the product of messages $m^{(1)}$ and $m^{(2)}$.

1. Using (7.39), set $tmp \leftarrow Dec_{sk}(C\_K\_m1\_2) = M^{(1)} \cdot M^{(2)} = K^2 \cdot m^{(1)} \cdot m^{(2)}$.
2. Set $m' \leftarrow tmp \cdot (K^{-1})^2 \ mod \ p = m^{(1)} \cdot m^{(2)}$.
3. **Return** m'

Note that Algorithm 7.1 outputs, mK', the product $(m^{(1)} \cdot m^{(2)})$ encrypted with K^2, while Algorithm 7.2, retrieves the product $(m^{(1)} \cdot m^{(2)})$ out of mK'.

Example I.6 in Appendix I is an example of recrypting the product of two ciphertexts.

In the next section computation of exponentiation homomorphically using recryption is shown.

7.2.2.3 Computing Exponentiation Homomorphically Using Recryption

In this section, we introduce computing plaintext, m, exponent, m^e, where $e = 2^n$, homomorphically using ciphertext c of m by Algorithm 7.3 and Algorithm 7.4 below. Algorithm 7.3 (Power2Exponent) calculates ciphertext (7.5) of M^e, where M is the padded message of m. Algorithm 7.4 (DecPower2Exponent) gets m^e from the output of Algorithm 7.3.

Algorithm 7.3:	Power2Exponent: Algorithm of homomorphic plaintext exponentiation M^{pwr} with $pwr = 2^n$ using ciphertext c of M
Input:	c, an RLWE-CSCM ciphertext of the message M; $pwr = 2^n$, exponent; k_{rec} recryption key (7.42) with recryption constant $K \in \mathbb{Z}_p$; pk, public key of RLWE-CSCM; r_rec, random polynomial used in encryption (7.5).

Output: C, an RLWE-CSCM ciphertext of M^{pwr}.

1. Set $C \leftarrow c$.
2. $while\ (pwr > 1)$
3. Calculate $tmp = M(C, C) \in R_q^3$ according to (7.23).
4. Set $Dec\_tmp \leftarrow Dec_{k_{rec}}(tmp)$ according to (7.43).
5. Set $C \leftarrow Enc_{pk,r\_rec}(\ Dec\_tmp) \in R_q^2$, where $Enc_{pk,r\_rec}(\ Dec\_tmp)$ is performed according to (7.5).
6. Set $pwr \leftarrow \frac{pwr}{2}$.
7. End while
8. **Return** C.

Algorithm 7.4 (DecPower2Exponent) describes the process of retrieving m^{pwr} from C obtained by Algorithm 7.3.

Algorithm 7.4: DecPower2Exponent: describes the process of retrieving m^{pwr} from C obtained by Algorithm 7.3

Input: c, an RLWE-CSCM ciphertext of the message M; $pwr = 2^n$, exponent; k_{rec} recryption key (7.42) with recryption constant $K \in \mathbb{Z}_p$; pk, public key of RLWE-CSCM; r_rec, random polynomial used in encryption (7.5).

Output: C, an RLWE-CSCM ciphertext of M^{pwr}.

1. Set $C \leftarrow c$.
2. $while\ (pwr > 1)$
3. Calculate $tmp = M(C, C) \in R_q^3$ according to (7.23).
4. Set $Dec\_tmp \leftarrow Dec_{k_{rec}}(tmp)$ according to (7.43).
5. Set $C \leftarrow Enc_{pk,r\_rec}(\ Dec\_tmp) \in R_q^2$, where $Enc_{pk,r\_rec}(\ Dec\_tmp)$ is performed according to (7.5).
6. Set $pwr \leftarrow \frac{pwr}{2}$.
7. End while
8. **Return** C.

Figure I.8 and Figure I.9 of Appendix I show Maple code implementation of Algorithm 7.3 Power2Exponent, and Algorithm 7.4 DecPower2Exponent respectively.

Example I.7: Example of calculating 512 multiplications homomorphically of $m^{(1)}$ in Example I.6 using Power2Exponent. in Appendix I shows the calculation of m^{512} homomorphically using Power2Exponent, for $m = 4x^2 + 5x + 1$.

To compute an exponent, pwr, that is not a power of two, the exponent pwr can be represented in binary. For example, to compute c^{1000}, the power, $pwr = 1000 = 512 + 256 + 128 + 64 + 32 + 8 = 1111101000_2$, therefore c^{1000} can be computed as

$$c^{1000} = c^{512} \cdot c^{256} \cdot c^{128} \cdot c^{64} \cdot c^{32} \cdot c^{8}$$

The values, $c^{512}, c^{256}, c^{128}, c^{64}, c^{32}, c^{8}$ can be found using Power2Exponent.

7.2.2.4 Homomorphic Multiplication Using Re-Linearization

The main idea of ciphertext re-linearization is to re-encrypt the product (7.23) under a new secret key, rsk, defined in (7.46), so that the original ciphertext form in (7.5) will be obtained.

$$rsk = (1, rs), \tag{7.46}$$

where rpk, the corresponding public key is defined in (7.47)

$$rpk = (rA \cdot rs + p \cdot re, -rA) = (rpk1, rpk2). \tag{7.47}$$

The steps of ciphertext re-linearization follow:

A- Encrypt the variable parts of the secret key, $sk3$, from (7.25)

$$cs = (rpk1 \cdot r3 + s, rpk2 \cdot r3) = (cs0, cs1), \tag{7.48}$$

$$cs2 = (rpk1 \cdot r4 + s^2, rpk2 \cdot r4) = (cs20, cs21)., \tag{7.49}$$

where $r3$, $r4$ are random variables. From (7.46)−(7.49)

$$se = s + p \cdot re = cs_0 + cs_1 \cdot rs, \tag{7.50}$$

$$s2e = s^2 + p \cdot re = cs2_0 + cs2_1 \cdot rs. \tag{7.51}$$

Recall from (7.26),(7.28),

$$m^{(1)} \cdot m^{(2)} = \langle M(c1, c2), sk3 \rangle \; mod \; p = \left\langle c^{(1)}, sk \right\rangle \cdot \left\langle c^{(2)}, sk \right\rangle mod \; p$$

$$= c_1^{(1)} c_1^{(2)} + \left(c_1^{(1)} c_2^{(2)} + c_2^{(1)} c_1^{(2)} \right) s + c_2^{(1)} c_2^{(2)} s^2 \; mod \; p$$

$$= (cm_0 + cm_1 \cdot s + cm_2 \cdot s^2) \; mod \; p \tag{7.52}$$

B- Substituting (7.50),(7.51) into (7.52), one gets

$$m1 \cdot m2 = < cm \cdot sk3 > \; mod \; p$$

$$= (cm_0 + cm_1 \cdot s + cm_2 \cdot s^2) \; mod \; p$$

$$= (cm_0 + cm_1 \cdot se + cm_2 \cdot s2e) \; mod \; p$$

$$= \left(cm_0 + cm_1 \cdot (cs_0 + cs_1 \cdot rs) + cm_2 \cdot (cs2_0 + cs2_1 \cdot rs) \right) mod \; p$$

$$= (cm_0 + cm_1 \cdot cs_0 + cm_2 \cdot cs2_0 + (cm_1 \cdot cs_1 + cm_2 \cdot cs2_1)$$

$$\cdot rs) \; mod \; p$$

$$= (crs_0 + crs_1 \cdot rs) \; mod \; p = < crs \cdot rsk > mod \; p \tag{7.53}$$

From (7.53), we get that ciphertext re-linearization of the product, $m1 \cdot m2$, is

$$(crs_0, crs_1) = (cm_0 + cm_1 \cdot cs_0 + cm_2 \cdot cs2_0, cm_1 \cdot cs_1 + cm_2$$

$$\cdot cs2_1) \tag{7.54}$$

which is decrypted using

$$m1 \cdot m2 = (crs_0 + crs_1 \cdot rs) \; mod \; p = < crs \cdot rsk > mod \; p \tag{7.55}$$

if

$$rs = s, \tag{7.56}$$

then the ciphertext (7.54) can be decrypted with the original secret key (7.4).

7.3 Security Analysis

7.3.1 Ciphertext Only Attack Against RLWE-CSCM Messages (Modulo C_2 Attack)

Modulo c_2 Attack can be launched against RLWE-CSCM messages if the conditions (7.57) and (7.58) are satisfied

$$\deg(pk_1 \cdot r + m) < d, \tag{7.57}$$

and,

$$\deg(c_2) > \deg(m)\,. \tag{7.58}$$

If (7.57) is satisfied, then, from (7.5),

$$c_1 = pk_1 \cdot r + m \; mod \; (x^d + 1) \; mod \; q$$

$$= pk_1 \cdot r + m \; mod \; q.$$

$$= pk_1 \cdot r + m - k \cdot \; q. \tag{7.59}$$

And if (7.58) is satisfied, then, applying $mod\; c_2$ operation to the message m, returns exact value of m. Therefore, if (7.57) and (7.58) are satisfied, the attacker can perform the following two steps of the attack:

Step 1: From (7.3), (7.5), (7.57), (7.58), applying $mod\; c_2$ operation to c_1 we have

$$S_1 = c_1 \; mod \; c_2 = A \cdot s \cdot r + p \cdot e \cdot r + m - k \cdot \; q \; mod \; (-A \cdot r)$$

$$= p \cdot e \cdot r + m \; - k \cdot \; q. \tag{7.60}$$

Step2: Applying $mod\; p$ operation to (7.60), and recalling (7.1), one gets,

$$m' = S_1 \; mod \; p \; = p \cdot e \cdot r + m \; - k \cdot \; q \; mod \; p = m. \tag{7.61}$$

Example I.8 of Appendix I is an example of a successful modulo c_2 attack. To counter this attack, it is enough to make sure one of the conditions (7.57) and (7.58) are not satisfied. Fulfilling constraint (7.6) violates condition (7.58). Example I.9 of Appendix I is an example of failing modulo c_2 attack.

7.3.2 Ciphertext Only Attack Against RLWE-CSCM Messages (Modulo p Attack)

Modulo p attack can be launched against RLWE-CSCM messages if the following condition is satisfied

$$\forall i \in \{0,..,d-1\}: a_i = \alpha_i \cdot p, \tag{7.62}$$

where a_i is a coefficient of polynomial $A \cdot s \cdot r \in R_q$, and $\alpha_i \in \mathbb{Z}^+$, i.e., the attack can be launched if all the coefficients of $A \cdot s \cdot r$ are multiple of p. In such case, from (7.3) and (7.5), one gets,

$$c_1 = \sum_{i=0}^{d-1}(a_i + b_i + m_i)x^i = \sum_{i=0}^{d-1}(\gamma_i \cdot p + m_i)x^i,$$

where b_i, m_i are coefficients of polynomials $p \cdot e \cdot r$, and m respectively. Thus, applying $mod\ p$ operation to c_1 retrieves the message. Since $a_i \in R_q$, and according to (7.1), there are α numbers multiple of p in R_q.therefore, the probability of satisfying (7.62) is $(\alpha/q)^d = (\alpha/(\alpha \cdot p))^d = (1/p)^d$

7.3.3 Ciphertext Only Attack Against RLWE-CSCM Messages Using Public Key

The encrypted message, m, can be compromised using the public key as follows,

Step1: Obtain random, r, from c_2,

$$r = A^{-1} \cdot (-c_2)\ mod\ q = (A^{-1} \cdot A \cdot r)\ mod\ q, \tag{7.63}$$

where A^{-1} is the multiplicative inverse of A' in R.

Step2: Revealing message, m, from c_1,

$$m = c_2 - pk_1 \cdot r. \tag{7.64}$$

This attack can be mitigated if the public key part, A, has no multiplicative inverse in R. This can be guaranteed by setting A meetings (7.2).

7.4 Summary

In this chapter, RLWE-CSCM is proposed, the first FHE scheme homomorphic w.r.t to two arithmetic operations and doesn't need NCM. It is based on RLWE and uses polynomials from the ring $R = \mathbb{Z}[x]/(x^d + 1)$. Section 7.1.5 shows that RLWE-CSCM decryption uses two moduli q, p, where the former is a multiple of the latter. Section 7.2.1 proves that applying modulo q operation in the first step of decryption, leaves the noise results from homomorphic addition(s) as a multiple of p, which vanishes by applying modulo p operation in the second step. Thus, the reason RLWE-CSCM is not affected by noise growth is the definition of moduli q in (7.1).

ciphertext results from the product of two ciphertexts (7.23) is greater than the original RLWE-CSCM ciphertext (7.5) by one. To avoid the increase of the ciphertext size, two CSCM's; ciphertext recryption; ciphertext linearization are proposed in Subsections 7.2.2.2, 7.2.2.4 respectively.

In ciphertext recryption, the product of two ciphertexts is decrypted using recryption key, k_rec, that results from the product of two plaintexts encrypted by multiplication with a secret recryption constant, K.

In ciphertext re-linearization, the product of two ciphertexts is encrypted again under a new secret key, rsk in (7.46), so that the form of the original ciphertext (7.5) is retrieved. In the case rsk in (7.46) is defined to be equal to sk in (7.4), then the re-linearized ciphertext (7.54) can be decrypted with the original secret key (7.4).

Three different COA against RLWE-CSCM messages have been defined in Section 7.3. It is also shown that RLWE-CSCM settings (7.2), (7.6), and (7.7) counter these attacks.

RLWE-CSCM is the first HE scheme not affected by noise growth, which allows performing an unlimited number of homomorphic additions and multiplications on the cihpertext. RLWE-CSCM can be used efficiently to protect data privacy in untrusted environment such as cloud. RLWE-CSCM can be used to provide security for various fields such as distance learning and medical applications.

Chapter 8

CONCLUSION

In this thesis, new HE ciphers classification is proposed. Contrary to all previously classifications proposed in (Domingo-Ferrer et al., 2019; Feng et al., 2020; Martins et al., 2017; Shrestha & Kim, 2019; Sultan, 2019; L. Wang & Ahmad, 2016; Zhao et al., 2020) it has the capability to accommodate new cryptosystems with new features. Therefore, it allows having a separate class for the newly developed herein RLWE-CSCM. The new HE classification introduced extends the previously used two criteria (homomorphic operation type, the permitted number of homomorphic operations) to five, adding: use of ciphertext size control mechanism; the number of keys used (symmetric, asymmetric); and the underlying hard problem. The proposed classification presented in Table 2.1, grouped known HE ciphers in eight classes.

RSA is the first developed PKC, it can provide multiplication operation homomorphically. Therefore, RSA is widely used in the Internet to prevent weak authentication, and in the public key certificates (Housley et al., 2002; Pandey et al., 2020, p. 321; Yakubu et al., 2019, p. 226). Many applications also involve the RSA multiplicative homomorphism feature such as secure image sharing (Islam et al. 2011), and homomorphic signatures (R. Johnson et al. 2002; Freeman 2012). Security analysis for RSA (Rivest et al., 1978) is considered in Chapter 3. We managed to formulate RSA encryption (2.2) to a 2-dimentional lattice (3.5). A new COA using LLL algorithm is proposed, which finds the encrypted message meeting (3.2)-(3.6), as

a component of the shortest vector in the lattice (3.5). The proposed COA has an advantage over previously proposed attacks against RSA, that it does not require prior knowledge of a number of bits; a small value of exponent e; message to be broadcasted (see Table 3.1). Experiments shown in Section 3.5, managed to reveal thousands of encrypted messages for N with bit sizes up to 8193 in Maple 2016.2 with success rate 0.1. Results presented in Table 3.3 show 80692 cracked messages for $N = 2050$ using Code 3.1. LLL attack runs in time quadratic in the bit number of modulo N (see Section 3.4). Our attack shows significant speed (15 milliseconds using Mupad, and 4×10^{-5} seconds using NTL (*NTL: A Library for Doing Number Theory*, n.d.) library for Example 3.1) in recovering a 40-bit message in comparison to our implementation for Boneh MITM attack (Boneh et al., 2000) where 2.202 seconds are needed to recover the same length message (2 seconds for pre-computation step, and 0.202 seconds for searching step using NTL library). Experiments show that proposed attack succeeds in the case when RSA public keys meet (3.16)-(3.18). Hence, to prevent proposed COA attack, it is important to select RSA public keys such that (3.16)-(3.18) are not satisfied.

NTRU is standardized as IEEE P1363.1, and it has been selected as a finalist in the NIST PQC standardization effort. Security of NTRU, representative of Class 3, is considered in Chapter 4. Modulo p flaw attack is designed, exploits a flaw found that in case (4.10) is satisfied, the encrypted message can be revealed without the need for the decryption key. The success probability of modulo p attack, $O(N^{2d+1})$, is derived in (4.38) and found to be not negligible using IEEE standardized parameters ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard

Problems over Lattices," 2009, p. 55). On other hand, the success probability becomes negligible by fixing the parameter $d = \left\lfloor \frac{N}{2} \right\rfloor$ as in (Howgrave-Graham et al., 2005).

Experiments conducted in Section 4.2 to test $IN - Lattice$ (Z. Yang et al., 2018b) attack on NTRU private keys use the settings in Table 4.2. They showed contrary to results in (Z. Yang et al., 2018b), the exponential growth of parameter t as N increases when $IN - Lattice$ attack succeeds (see Figure 4.3). It means that the success probability of the attack is low for large value of N. For $N = 107$, we failed to attack NTRU private key using $IN - Lattice$ attack after 6 hours. Experimental data were approximated using quadratic fitting in (4.40) instead of linear fitting presented in (Z. Yang et al., 2018b). Thus, extrapolation of time for greater N values presented in (4.41) using a quadratic approximation in (4.40), and shown in Table 4.4, has greater time than extrapolation line in (Z. Yang et al., 2018a). Therefore, we conclude that the attack is not so efficient as claimed, and the attack is infeasible for large values of N.

To counter LBRA using LLL, NTRU and its variants are shown in section 2.3.4 uses high order N, which increases the computational complexity. To solve this issue, RCPKC, which is presented in Chapter 6, uses polynomials of zero degrees, that is integer. RCPKC is a secure variant of insecure CPKC presented by the authors of NTRU as a toy model (Hoffstein et al., 2014a). The insecurity of CPKC stems from the choice of the private keys used as small numbers to provide decryption correctness. Thus, it is prone to LBRA using GLR (crackable in about 10 iterations) (see Example C.2 in Appendix C). RCPKC specifies a range from which the random numbers shall be selected and provides correct decryption for valid users and incorrect decryption for an attacker using LBRA by GLR. Security of RCPKC is proved against various

attacks in Section 6.3. In Section 6.4, RCPKC asymmetric encryption padding (RAEP), is proposed. RAEP similar to its NTRU analog, NAEP, is IND-CCA2 secure. Due to the use of big numbers instead of high degree polynomials, RCPKC is about 27 times faster in encryption and decryption than NTRU. Furthermore, RCPKC is more than three times faster than the most effective known NTRU variant, BQTRU. Compared to NTRU, RCPKC reduces energy consumption at least thirty times, which allows increasing the life-time of unattended WSNs more than thirty times. RCPKC performance and power analysis are conducted in Section 6.5.

In Chapter 5, the security of HE1N (Dyer et al., 2019), representative of Class 4, is considered. Section 5.1 proves that the modulus operation used in HE1N encryption, Algorithm 2.4, is not working, which leaves HE1N private key prone to various attacks. In sections 5.2.1 and 5.3.1 a new COA and KPA against HE1N private key are presented respectively. In sections 5.2.3 and 5.3.3 it is proved that the success probability of both attacks becomes negligible as the length of parameter k in bits increases. The computational complexity of COA and KPA is $O(m^2)$. HE1N is one of several cryptosystems that are proposed in (Dyer et al., 2019) as a practical solution to provide security for clouds. Therefore, it is important to select parameters so that the success probability of both KPA and COA become negligible.

From Table 2.1, we can see that all HE schemes support two homomorphic schemes that need NCM to overcome the issue of noise growth due to increasing executed homomorphic operations. In Chapter 7, RLWE-CSCM, the first FHE that is not affected by noise growth by construction, is proposed. It is based on RLWE and uses polynomials in $R = \mathbb{Z}[x]/(x^d + 1)$. Homomorphism of RLWE-CSCM with respect to addition and multiplication is proved in Section 7.2. Ciphertext size of RLWE-

CSCM increases by executing multiplication operation. Therefore, CSCM is needed. Two CSCMs are proposed; ciphertext recryption and ciphertext re-linearization in sections 7.2.2.2 and 7.2.2.4 respectively. Security of RLWE-CSCM against various attacks is presented in Section 7.3. Experiments by (Gentry & Halevi, 2011) showed that bootstrapping NCM consumes significant time, and according to (Sarkar et al., 2021, p. 133,134), the implementations of such encryption schemes remain unsuitable for real-time applications yet. Therefore, proposed an FHE without NCM can be involved in many real-time applications.

The practical results of the thesis work are:

1. A new HE ciphers classification is proposed. Contrary to all previous HE ciphers classifications; it has the capability to accommodate new cryptosystems with new features. Hence, it allows having a separate class for the newly developed herein HE cipher, RLWE-CSCM.
2. The proposed COA attack against RSA compromises messages encrypted with large modulus greater than 8000 bits efficiently. Proposed attack threatens users of RSA on Internet and applications exploiting homomorphic multiplicative feature of RSA. The attack can be avoided by having RSA public key to violate conditions (3.16)-(3.18).
3. NTRU modulo p flaw attack is proposed exploiting a flaw that in case of (4.10) is satisfied, the encrypted message can be revealed without the need for the private key. The attack threatens the users of NTRU with IEEE standard parameters in ("IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices," 2009, p. 55) with non-negligible success probability. To make the success probability of NTRU

modulo p attack negligible, users are recommended to set the parameter $d = \left\lfloor \frac{N}{2} \right\rfloor$ as in (Howgrave-Graham et al., 2005).

4. RCPKC, the random congruential public key cryptosystem is developed. RCPKC is resistant to LBRA using GLR. It is more secure and efficient than NTRU and all its variants. Therefore, it can run on devices with constrained computational capability, and devices with constrained power, such as unattended WSNs.
5. RLWE-CSCM, ring learning with errors with ciphertext size control mechanism is proposed. RLWE-CSCM is the first HE cipher not affected by noise growth due to homomorphic operations. Therefore, RLWE-CSCM is a practical solution to provide data privacy for users rely on cloud computing services.

As a future work, RLWE-CSCM performance analysis will be conducted, a more extensive RLWE-CSCM security analysis will be executed, a hardware implementation of both RCPKC and RLWE-CSCM will be made, and both of RCPKC and RLWE-CSCM will be applied to telemedicine to secure the data collected by medical sensors and cameras.

Finally, I would like to mention that the publication requirement for the Ph.D. degree of having at least one conference paper, and one SCI-Expanded journal paper are fulfilled with the following published papers: (Chefranov & Ibrahim, 2016; Easttom et al., 2020; Ibrahim et al., 2020, 2021, 2019; Ibrahim & Chefranov, 2016).

REFERENCES

Abd Ghafar, A. H., Kamel Ariffin, M. R., & Asbullah, M. A. (2020). A New LSB Attack on Special-Structured RSA Primes. *Symmetry*, *12*(5). https://doi.org/10.3390/sym12050838

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Computing Surveys*, *51*(4), 1–35. https://doi.org/10.1145/3214303

Alimoradi, R., & Arkian, H. R. (2016). Integer Factorization Implementations. *ICTACT Journal on Communication Technology*, *7*(2), 1310–1314. https://doi.org/10.21917/ijct.2016.0192

anasnet/RCPKC-Project: Open Source RCPKC Cryptosystem Reference Code. (n.d.). Retrieved June 30, 2021, from https://github.com/anasnet/RCPKC-Project

Bagheri, K., Sadeghi, M.-R., & Panario, D. (2018). A Non-commutative Cryptosystem Based on Quaternion Algebras. *Designs, Codes and Cryptography*, *86*(10), 2345–2377. https://doi.org/10.1007/s10623-017-0451-4

Banks, W. D., & Shparlinski, I. (2002). A Variant of NTRU with Non-Invertible Polynomials. In A. Menezes & P. Sarkar (Eds.), *Proceedings of the Third International Conference on Cryptology: Progress in Cryptology* (Vol. 2551, pp. 62–70). Springer-Verlag. https://doi.org/10.1007/3-540-36231-2_6

Barker, E. (n.d.). *NIST Special Publication 800-57 Part 1 Revision 5 Recommendation for Key Management: Part 1-General*. Retrieved October 25, 2021, from https://doi.org/10.6028/NIST.SP.800-57pt1r5

Barker, E., & Dang, Q. (n.d.). *Recommendation for Key Management Part 3: Application-Specific Key Management Guidance*. Retrieved October 25, 2021, from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 3279 RFC 1 (2002). https://doi.org/10.17487/RFC3279

Belhaj, S., & Kahla, H. ben. (2013). On the Complexity of Computing the GCD of Two Polynomials Via Hankel Matrices. *ACM Communications in Computer Algebra*, *46*(3–4), 74–75. https://doi.org/10.1145/2429135.2429140

Beloglazov, A., Buyya, R., Lee, Y. C., & Zomaya, A. (2011). A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Advances in Computers*, *82*, 47–111. https://doi.org/10.1016/B978-0-12-385512-1.00003-7

Benabbes, S., & Hemam, S. M. (2019). An Approach Based on (Tasks-VMs) Classification and MCDA for Dynamic Load Balancing in the CloudIoT. *Lecture Notes in Networks and Systems*, *102*, 387–396. https://doi.org/10.1007/978-3-030-37207-1_41

Berkovits, S. (1982). Factoring via Superencyrption. *Cryptologia*, *6*(3), 229–237. https://doi.org/10.1080/0161-118291857028

Bleichenbacher, D. (1997). On the Security of the KMOV Public Key Cryptosystem. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1294*, 235–248. https://doi.org/10.1007/BFB0052239

Boneh, D., & Durfee, G. (2000). Cryptanalysis of RSA with Private Key d Less than N^0.292. *IEEE Trans. Inf. Theory*, *46*(4), 1339–1349. https://doi.org/10.1109/18.850673

Boneh, D., & Durfee, G. (1999). Cryptanalysis of RSA with Private Key d Less than N^0.292. In J. Stern (Ed.), *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (Vol. 1592, pp. 1–11). Springer. https://doi.org/10.1007/3-540-48910-X_1

Boneh, D., Durfee, G., & Franke, Y. (1998). An Attack on RSA Given a Small Fraction of the Private Key Bits. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1514*, 25–34. https://doi.org/10.1007/3-540-49649-1_3

Boneh, D., Joux, A., & Nguyen, P. Q. (2000). Why Textbook Elgamal and RSA Encryption are Insecure: (Extended Abstract). *Lecture Notes in Computer*

Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *1976*, 30–43. https://doi.org/10.1007/3-540-44448-3_3

Boulemtafes, A., Derhab, A., Braham, N. A. A., & Challal, Y. (2021). Privacy-Preserving Remote Deep-Learning-Based Inference Under Constrained Client-Side Environment. *Journal of Ambient Intelligence and Humanized Computing 2021*, 1–14. https://doi.org/10.1007/S12652-021-03312-8

Brakerski, Z. (2019). Fundamentals of Fully Homomorphic Encryption. In O. Goldreich (Ed.), *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali* (pp. 543–563). ACM. https://doi.org/10.1145/3335741.3335762

Brakerski, Z., Gentry, C., & Halevi, S. (2013). Packed Ciphertexts in LWE-Based Homomorphic Encryption. In K. Kurosawa & G. Hanaoka (Eds.), *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings* (Vol. 7778, pp. 1–13). Springer. https://doi.org/10.1007/978-3-642-36362-7_1

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) Fully Homomorphic Encryption Without Bootstrapping. *ACM Transactions on Computation Theory*, *6*(3), 1–36. https://doi.org/10.1145/2633600

Brakerski, Z., & Vaikuntanathan, V. (2011a). Efficient Fully Homomorphic Encryption from (Standard) LWE. In R. Ostrovsky (Ed.), *IEEE 52nd Annual*

Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011 (pp. 97–106). IEEE Computer Society. https://doi.org/10.1109/FOCS.2011.12

Brakerski, Z., & Vaikuntanathan, V. (2011b). Fully homomorphic encryption from ring-LWE and security for key dependent messages. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6841 LNCS*, 505–524. https://doi.org/10.1007/978-3-642-22792-9_29

Bunder, M., Nitaj, A., Susilo, W., & Tonien, J. (2017). A Generalized Attack on RSA Type Cryptosystems. *Theoretical Computer Science*, *704*, 74–81. https://doi.org/10.1016/j.tcs.2017.09.009

[Cado-nfs-discuss] Factorization of RSA-250. (n.d.). Retrieved June 1, 2021, from https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2020-February/001166.html

Chefranov, A., & Ibrahim, A. (2016). NTRU Modulo p Flaw. *World Congress on Internet Security (WorldCIS-2016)*, 54–58. https://infonomics-society.org/worldcis-2016/worldcis-abstract-10/

Chen, L., Ben, H., & Huang, J. (2014). An Encryption Depth Optimization Scheme for Fully Homomorphic Encryption. *Proceedings - 2014 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2014*, 137–141. https://doi.org/10.1109/IIKI.2014.35

Cheon, J. H., Coron, J. S., Kim, J., Lee, M. S., Lepoint, T., Tibouchi, M., & Yun, A. (2013). Batch Fully Homomorphic Encryption over the Integers. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7881*, 315–335. https://doi.org/10.1007/978-3-642-38348-9_20

Chuang, Y. L., Fan, C. I., & Tseng, Y. F. (2018). An Efficient Algorithm for the Shortest Vector Problem. *IEEE Access*, *6*, 61478–61487. https://doi.org/10.1109/ACCESS.2018.2876401

Coglianese, M., & Goi, B.-M. (2005). MaTRU: A New NTRU-Based Cryptosystem. In S. Maitra, C. E. V. Madhavan, & R. Venkatesan (Eds.), *Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings* (Vol. 3797, pp. 232–243). Springer. https://doi.org/10.1007/11596219_19

Coppersmith, D. (1996a). Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1070*, 178–189. https://doi.org/10.1007/3-540-68339-9_16

Coppersmith, D. (1996b). Finding a Small Root of a Univariate Modular Equation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1070*, 155–165. https://doi.org/10.1007/3-540-68339-9_14

Coppersmith, D., Franklin, M. K., Patarin, J., & Reiter, M. K. (1996). Low-Exponent RSA with Related Messages. In U. M. Maurer (Ed.), *Advances in Cryptology - {EUROCRYPT} '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding* (Vol. 1070, pp. 1–9). Springer. https://doi.org/10.1007/3-540-68339-9_1

Coron, J. S., Mandal, A., Naccache, D., & Tibouchi, M. (2011). Fully Homomorphic Encryption over the Integers with Shorter Public Keys. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6841*, 487–504. https://doi.org/10.1007/978-3-642-22792-9_28

Coron, J. S., Naccache, D., & Tibouchi, M. (2012). Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7237*, 446–464. https://doi.org/10.1007/978-3-642-29011-4_27

de Weger, B. (2002). Cryptanalysis of RSA with Small Prime Difference. *Applicable Algebra in Engineering, Communications and Computing*, *13*(1), 17–28. https://doi.org/10.1007/s002000100088

Delaurentis, J. M. (1984). A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm. *Cryptologia*, *8*(3), 253–259. https://doi.org/10.1080/0161-118491859060

Diaconis, P., & Mosteller, F. (1989). Methods for studying coincidences. *Journal of the American Statistical Association*, *84*(408), 853–861. https://doi.org/10.1080/01621459.1989.10478847

Domingo-Ferrer, J., Farràs, O., Ribes-González, J., & Sánchez, D. (2019). Privacy-preserving Cloud Computing on Sensitive Data: A Survey of Methods, Products and Challenges. In *Computer Communications* (Vols. 140–141, pp. 38–60). Elsevier B.V. https://doi.org/10.1016/j.comcom.2019.04.011

Doröz, Y., Hoffstein, J., Pipher, J., Silverman, J. H., Sunar, B., Whyte, W., & Zhang, Z. (2018). Fully Homomorphic Encryption from the Finite Field Isomorphism Problem. In M. Abdalla & R. Dahab (Eds.), *Public-Key Cryptography – PKC 2018. Lecture Notes in Computer Science, vol 10769.* (pp. 125–155). Springer International Publishing. https://doi.org/https://doi.org/10.1007/978-3-319-76578-5_5

Doröz, Y., & Sunar, B. (2020). Flattening NTRU for Evaluation Key Free Homomorphic Encryption. *Journal of Mathematical Cryptology*, *14*(1), 66–83. https://doi.org/10.1515/jmc-2015-0052

Dyer, J., Dyer, M., & Xu, J. (2019). Practical Homomorphic Encryption over the Integers for Secure Computation in the Cloud. *International Journal of Information Security*, *18*(5), 549–579. https://doi.org/10.1007/s10207-019-00427-0

Easttom, C., Ibrahim, A., Chefranov, A., Alsmadi, I., & Hansen, R. (2020). Towards A Deeper NTRU Analysis: A Multi Modal Analysis. *International Journal on Cryptography and Information Security (IJCIS)*, *10*(2), 11–22. https://doi.org/10.5121/ijcis.2020.10202

ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *196*, 10–18. https://doi.org/10.1007/3-540-39568-7_2

Feng, J., Yang, L. T., Gati, N. J., Xie, X., & Gavuna, B. S. (2020). Privacy-Preserving Computation in Cyber-Physical-Social Systems: A Survey of the State-of-the-Art and Perspectives. *Information Sciences*, *527*, 341–355. https://doi.org/10.1016/j.ins.2019.07.036

Freeman, D. M. (2012). Improved Security for Linearly Homomorphic Signatures: A Generic Framework. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7293*, 697–714. https://doi.org/10.1007/978-3-642-30057-8_41

Gaborit, P., Ohler, J., & Solé, P. (2002). *CTRU, a polynomial analogue of NTRU*. INRIA. https://hal.inria.fr/inria-00071964

Gaithuru, J. N., & Salleh, M. (2017). ITRU: NTRU-Based Cryptosystem Using Ring of Integers. *International Journal of Innovative Computing*, *7*(1). https://doi.org/10.11113/ijic.v7n1.135

Gama, N., & Nguyen, P. Q. (2007). New Chosen-Ciphertext Attacks on NTRU. In T. Okamoto & X. Wang (Eds.), *Public Key Cryptography -- PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16-20, 2007. Proceedings* (pp. 89–106). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71677-8_7

Gao, S. (2018). Efficient Fully Homomorphic Encryption Scheme. *IACR Cryptol. EPrint Arch.*, 637. https://eprint.iacr.org/2018/637

Gentry, C. (2009a). *A FULLY HOMOMORPHIC ENCRYPTION SCHEME*. https://crypto.stanford.edu/craig/craig-thesis.pdf

Gentry, C. (2010). Computing Arbitrary Functions of Encrypted Data. *CommunICAtIonS of the ACm*, *53*(3). https://doi.org/10.1145/1666420.1666444

Gentry, C. (2009b). Fully Homomorphic Encryption Using Ideal Lattices. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 169–178. https://doi.org/10.1145/1536414.1536440

Gentry, C., & Halevi, S. (2011). Implementing Gentry's Fully-Homomorphic Encryption Scheme. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6632*, 129–148. https://doi.org/10.1007/978-3-642-20465-4_9

GitHub - NTRUOpenSourceProject/ntru-crypto: Open Source NTRU Public Key Cryptography and Reference Code. (n.d.). Retrieved June 30, 2021, from https://github.com/NTRUOpenSourceProject/ntru-crypto

Gorroochurn, P. (2012). Classic Problems of Probability. In *Classic Problems of Probability* (pp. 240–246). John Wiley & Sons, Inc. https://www.wiley.com/en-us/Classic+Problems+of+Probability-p-9781118063255

Halevi, S., & Shoup, V. (2021). Bootstrapping for HElib. *Journal of Cryptology*, *34*(1), 1–44. https://doi.org/10.1007/s00145-020-09368-7

Han, J., Kamber, M., & Pei, J. (2012). 2 - Getting to Know Your Data. In J. Han, M. Kamber, & J. Pei (Eds.), *Data Mining (Third Edition)* (pp. 39–82). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-381479-1.00002-2

Han, Z., Li, X., Huang, K., & Feng, Z. (2018). A Software Defined Network-Based Security Assessment Framework for CloudIoT. *IEEE Internet of Things Journal*, *5*(3), 1424–1434. https://doi.org/10.1109/JIOT.2018.2801944

Hastad, J. (1988). Solving Simultaneous Modular Equations of Low Degree. *SIAM Journal on Computing*, *17*(2), 336–341. https://doi.org/10.1137/0217019

Hastad, J. (1986). On Using RSA with Low Exponent in a Public Key Network. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *218*, 403–408. https://doi.org/10.1007/3-540-39799-X_29

Hermans, J., Vercauteren, F., & Preneel, B. (2010). Speed Records for NTRU. In J. Pieprzyk (Ed.), *Topics in Cryptology - CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings* (pp. 73–88). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-11925-5_6

Hoffstein, J., Howgrave-Graham, N., Pipher, J., & Whyte, W. (2010). Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign. In *Information Security and Cryptography* (Vol. 10, pp. 349–390). Springer International Publishing. https://doi.org/10.1007/978-3-642-02295-1_11

Hoffstein, J., Pipher, J., Schanck, J. M., Silverman, J. H., Whyte, W., & Zhang, Z. (2017). Choosing Parameters for NTRUEncrypt. In H. Handschuh (Ed.), *Topics in Cryptology -- CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14--17, 2017, Proceedings* (pp. 3–18). Springer International Publishing. https://doi.org/10.1007/978-3-319-52153-4_1

Hoffstein, J., Pipher, J., & Silverman, J. H. (1998). NTRU: A Ring-Based Public Key Cryptosystem. In J. P. Buhler (Ed.), *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21--25, 1998 Proceedings* (pp. 267–288). Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0054868

Hoffstein, J., Pipher, J., & Silverman, J. H. (2014a). A Congruential Public Key Cryptosystem. In *An Introduction to Mathematical Cryptography* (pp. 373–376). Springer-Verlag New York. https://doi.org/10.1007/978-1-4939-1711-2

Hoffstein, J., Pipher, J., & Silverman, J. H. (2014b). *An Introduction to Mathematical Cryptography*. Springer New York. https://doi.org/10.1007/978-1-4939-1711-2

Hoffstein, J., Pipher, J., & Silverman, J. H. (2014c). Gaussian Lattice Reduction in Dimension 2. In *An Introduction to Mathematical Cryptography* (pp. 436–438). Springer-Verlag New York. https://doi.org/10.1007/978-1-4939-1711-2

Hoffstein, J., Silverman, J. H., & Whyte, W. (1999). Estimated Breaking Times for NTRU Lattices. *Version 2, NTRU Cryptosystems (2003) Http://Citeseerx.Ist.Psu.Edu/Viewdoc/Summary?Doi=10.1.1.10.6336*.

Housley, R., Polk, W., Ford, W., & Solo, D. (2002). *RFC3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC Editor.

Howgrave-Graham, N. (2007). A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4622*, 150–169. https://doi.org/10.1007/978-3-540-74143-5_9

Howgrave-Graham, N., Nguyen, P. Q., Pointcheval, D., Proos, J., Silverman, J. H., Singer, A., & Whyte, W. (2003). The impact of decryption failures on the security

of NTRU encryption. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2729*, 226–246. https://doi.org/10.1007/978-3-540-45146-4_14

Howgrave-Graham, N., Silverman, J. H., Singer, A., & Whyte, W. (2003). NAEP: Provable Security in the Presence of Decryption Failures. *IACR Cryptol. EPrint Arch.*, *2003*, 172. http://eprint.iacr.org/2003/172

Howgrave-Graham, N., Silverman, J. H., & Whyte, W. (2003). *A Meet-in-the-Middle Attack on an NTRU Private key*. Technical report, NTRU Cryptosystems, June 2003. Report. http://grouper.ieee.org/groups/1363/lattPK/submissions/NTRUTech004v2.pdf

Howgrave-Graham, N., Silverman, J. H., & Whyte, W. (2005). Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. *IACR Cryptol. EPrint Arch.*, *2005*, 45. http://eprint.iacr.org/2005/045

Ibrahim, A., & Chefranov, A. (2016). NTRU Modulo p Flaw. *International Journal for Information Security Research*, *6*(3). https://doi.org/10.20533/ijisr.2042.4639.2016.0079

Ibrahim, A., Chefranov, A., & Hamad, N. (2019). Ntru-like secure and effective congruential public-key cryptosystem using big numbers. *2019 2nd International Conference on New Trends in Computing Sciences, ICTCS 2019 - Proceedings*, 1–7. https://doi.org/10.1109/ICTCS.2019.8923091

Ibrahim, A., Chefranov, A., Hamad, N., Daraghmi, Y. A., Al-Khasawneh, A., & Rodrigues, J. J. P. C. (2020). Ntru-like random congruential public-key cryptosystem for wireless sensor networks. *Sensors (Switzerland)*, *20*(16), 4632. https://doi.org/10.3390/s20164632

Ibrahim, A., Chefranov, A., & Hamamreh, R. (2021). Ciphertext-only attack on RSA using lattice basis reduction. *International Arab Journal of Information Technology*, *18*(2), 237–247. https://doi.org/10.34028/IAJIT/18/2/13

IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. (2009). *IEEE Std 1363.1-2008*, 1–81. https://doi.org/10.1109/IEEESTD.2009.4800404

Islam, N., Puech, W., Hayat, K., & Brouzet, R. (2011). Application of homomorphism to secure image sharing. *Optics Communications*, *284*(19), 4412–4429. https://doi.org/10.1016/j.optcom.2011.05.079

Jamnig, P. (1988). Securing the RSA-Cryptosystem Against Cycling Attacks. *Cryptologia*, *12*(3), 159–164. https://doi.org/10.1080/0161-118891862891

Jarvis, K., & Nevins, M. (2015). ETRU: NTRU over the Eisenstein integers. *Des. Codes Cryptogr.*, *74*(1), 219–242. https://doi.org/10.1007/s10623-013-9850-3

Jaulmes, É., & Joux, A. (2000). A Chosen-Ciphertext Attack against NTRU. In M. Bellare (Ed.), *Advances in Cryptology --- CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August*

20--24, 2000 Proceedings (pp. 20–35). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44598-6_2

Joe, S. (2016). The Hermite normal form for certain rank-1 circulant and skew-circulant lattice rules. *Linear Algebra and Its Applications*, *496*, 438–451. https://doi.org/10.1016/j.laa.2016.02.009

Johnson, R., Molnar, D., Song, D., & Wagner, D. (2002). Homomorphic Signature Schemes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2271*, 244–262. https://doi.org/10.1007/3-540-45760-7_17

Kaliski, B., Jonsson, J., & Rusch, A. (2016). *RFC8017: PKCS #1: RSA Cryptography Specifications Version 2.2* (K. Moriarty, Ed.). https://doi.org/10.17487/RFC8017

Karbasi, A. H., & Atani, R. E. (2015). ILTRU: An NTRU-Like Public Key Cryptosystem Over Ideal Lattices. *IACR Cryptol. EPrint Arch.*, *2015*, 549. http://eprint.iacr.org/2015/549

Kelaidonis, D., Vlacheas, P., Stavroulaki, V., Georgoulas, S., Moessner, K., Hashi, Y., Hashimoto, K., Miyake, Y., Yamada, K., & Demestichas, P. (2017). Cloud Internet of Things Framework for Enabling Services in Smart Cities. In *Designing, Developing, and Facilitating Smart Cities* (pp. 163–191). Springer International Publishing. https://doi.org/10.1007/978-3-319-44924-1_9

Kirchner, P., & Fouque, P. A. (2017). Revisiting Lattice Attacks on Overstretched NTRU Parameters. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10210*, 3–26. https://doi.org/10.1007/978-3-319-56620-7_1

Kleinjung, T., Aoki, K., Franke, J., Lenstra, A. K., Thomé, E., Bos, J. W., Gaudry, P., Kruppa, A., Montgomery, P. L., Osvik, D. A., te Riele, H., Timofeev, A., & Zimmermann, P. (2010). Factorization of a 768-Bit RSA modulus. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6223*, 333–350. https://doi.org/10.1007/978-3-642-14623-7_18

Lenstra, A. K., Lenstra, H. W., & Lovász, L. (1982). Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, *261*(4), 515–534. https://doi.org/10.1007/BF01457454

Lenstra, A. K., & Verheul, E. R. (2000). Selecting cryptographic key sizes extended abstract. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1751*, 446–465. https://doi.org/10.1007/978-3-540-46588-1_30

Listserv - Nmbrthry Archives. (n.d.). Retrieved June 1, 2021, from https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;fd743373.1912&S=

López-Alt, A., Tromer, E., & Vaikuntanathan, V. (2012). On-the-Fly Multiparty Computation on the Cloud Via Multikey Fully Homomorphic Encryption. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 1219–1234. https://doi.org/10.1145/2213977.2214086

Malekian, E., & Zakerolhosseini, A. (2010). OTRU: A non-associative and high speed public key cryptosystem. *Proceedings - 15th CSI International Symposium on Computer Architecture and Digital Systems, CADS 2010*, 83–90. https://doi.org/10.1109/CADS.2010.5623536

Malekian, E., Zakerolhosseini, A., & Mashatan, A. (2011). QTRU: quaternionic version of the NTRU public-key cryptosystems. *ISC Int. J. Inf. Secur.*, *3*(1), 29–42. https://doi.org/10.22042/isecure.2015.3.1.3

Margaret Amala, S., & Gnana Jayanthi, J. (2020). Internet of Things: A Technical Perspective Survey. In *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications* (pp. 659–667). Springer International Publishing. https://doi.org/10.1007/978-3-030-24051-6_60

Martins, P., Sousa, L., & Mariano, A. (2017). A Survey on Fully Homomorphic Encryption: An Engineering Perspective. *ACM Comput. Surv.*, *50*(6). https://doi.org/10.1145/3124441

May, A. (2010). Using LLL-Reduction for Solving RSA and Factorization Problems. In *Information Security and Cryptography* (Vol. 10, pp. 315–348). Springer International Publishing. https://doi.org/10.1007/978-3-642-02295-1_10

M.G., N., & R., H. (2016). BITRU: Binary Version of the NTRU Public Key Cryptosystem via Binary Algebra. *International Journal of Advanced Computer Science and Applications*, *7*(11). https://doi.org/10.14569/ijacsa.2016.071101

Micciancio, D. (2016). Shortest Vector Problem. In *Encyclopedia of Algorithms* (pp. 1974–1977). Springer New York. https://doi.org/10.1007/978-1-4939-2864-4_374

Micciancio, D. (2001). Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In J. H. Silverman (Ed.), *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers* (Vol. 2146, pp. 126–145). Springer. https://doi.org/10.1007/3-540-44670-2_11

Moore, D. S. (2006). *The Basic Practice of Statistics* (4th ed.). W. H. Freeman.

Nikolić, B. (2008). Design in the power-limited scaling regime. *IEEE Transactions on Electron Devices*, *55*(1), 71–83. https://doi.org/10.1109/TED.2007.911350

NTL: A Library for doing Number Theory. (n.d.). Retrieved June 1, 2021, from https://libntl.org/

Nuida, K., & Kurosawa, K. (2015). (Batch) Fully Homomorphic Encryption over Integers for Non-Binary Message Spaces. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9056*, 537–555. https://doi.org/10.1007/978-3-662-46800-5_21

Pandey, P., Pandey, S. C., & Kumar, U. (2020). *Security Issues of Internet of Things in Health-Care Sector: An Analytical Approach* (pp. 307–329). Springer, Singapore. https://doi.org/10.1007/978-981-15-1100-4_15

PQC Third Round Candidate Announcement | CSRC. (n.d.). Retrieved May 29, 2021, from https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement

Pulido-Gaytan, B., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., Radchenko, G., Avetisyan, A., & Drozdov, A. Y. (2021). Privacy-Preserving Neural Networks with Homomorphic Encryption: Challenges and Opportunities. *Peer-to-Peer Networking and Applications*, *14*(3), 1666–1691. https://doi.org/10.1007/s12083-021-01076-8

Rabah, K. (2006). Review of Methods for Integer Factorization Applied to Cryptography. In *Journal of Applied Sciences* (Vol. 6, Issue 2, pp. 458–481). https://doi.org/10.3923/jas.2006.458.481

Rabie, A. H., Saleh, A. I., & Ali, H. A. (2021). Smart Electrical Grids Based on Cloud, IoT, and Big Data Technologies: State of The Art. *J. Ambient Intell. Humaniz. Comput.*, *12*(10), 9449–9480. https://doi.org/10.1007/s12652-020-02685-6

Ramaiah, Y. G., & Kumari, G. V. (2012). Efficient Public Key Homomorphic Encryption over Integer Plaintexts. *Proceedings - 3rd International Conference on Information Security and Intelligent Control, ISIC 2012*, 123–128. https://doi.org/10.1109/ISIC.2012.6449723

Ramesh, S., & Govindarasu, M. (2020). An Efficient Framework for Privacy-Preserving Computations on Encrypted IoT Data. *IEEE Internet of Things Journal*, *7*(9), 8700–8708. https://doi.org/10.1109/JIOT.2020.2998109

rfc3279. (n.d.). Retrieved May 28, 2021, from https://datatracker.ietf.org/doc/html/rfc3279#section-2.3.1

Rivest, R. L. (1978). Remarks on a Proposed Cryptanalytic Attack on the M.I.T. Public-Key Cryptosystem. *Cryptologia*, *2*(1), 62–65. https://doi.org/10.1080/0161-117891852785

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Romli, N. B., Minhad, K. N., I Reaz, M. B., & Amin, S. (2015). AN OVERVIEW OF POWER DISSIPATION AND CONTROL TECHNIQUES IN CMOS TECHNOLOGY. In *Journal of Engineering Science and Technology* (Vol. 10, Issue 3).

RSA Factoring Challenge - Wikipedia. (n.d.). Retrieved June 1, 2021, from https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Sarkar, A., Chatterjee, S. R., & Chakraborty, M. (2021). Role of Cryptography in Network Security. In *Lecture Notes in Networks and Systems* (Vol. 163, pp. 103–

143). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-981-15-9317-8_5

Seck, M., & Sow, D. (2019). *BI-NTRU Encryption Schemes: Two New Secure Variants of NTRU*. 216–235. https://doi.org/10.1007/978-3-030-36237-9_13

Sedenion - Wikipedia. (n.d.). Retrieved June 5, 2021, from https://en.wikipedia.org/wiki/Sedenion

Sethi, R., Bhushan, B., Sharma, N., Kumar, R., & Kaushik, I. (2021). *Applicability of Industrial IoT in Diversified Sectors: Evolution, Applications and Challenges* (pp. 45–67). Springer, Singapore. https://doi.org/10.1007/978-981-15-7965-3_4

Shah, J. L., & Bhat, H. F. (2020). CloudIoT for Smart Healthcare: Architecture, Issues, and Challenges. In *Internet of Things Use Cases for the Healthcare Industry* (pp. 87–126). Springer International Publishing. https://doi.org/10.1007/978-3-030-37526-3_5

Shrestha, R., & Kim, S. (2019). Integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities. In *Advances in Computers* (Vol. 115, pp. 293–331). Academic Press Inc. https://doi.org/10.1016/bs.adcom.2019.06.002

Silverman, J. H., Silverman, J. H., & Whyte, W. (2003). *Estimating Decryption Failure Probabilities for NTRUEncrypt*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.7016

Simmons, G. J. (1977). Preliminary Comments on the M.I.T. Public-Key Cryptosystem. *Cryptologia*, *1*(4), 406–414. https://doi.org/10.1080/0161-117791833219

Sinchana, M. K., & Savithramma, R. M. (2020). Survey on Cloud Computing Security. In *Lecture Notes in Networks and Systems* (Vol. 103, pp. 1–6). Springer. https://doi.org/10.1007/978-981-15-2043-3_1

Sipser, M. (2013). *Introduction to the Theory of Computation* (M. Lee, Ed.; 3rd ed.). Cengage Learning. https://www.cengage.co.uk/books/9780357670583/

Smart, N. P., & Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6056 LNCS*, 420–443. https://doi.org/10.1007/978-3-642-13013-7_25

Smith, C. (2002). *Environmental Physics* (1st Edition). Routledge Book. https://www.routledge.com/Environmental-Physics/Smith/p/book/9780415201919

Stallings, W. (2014). Fermat's and Euler's Theorem. In *Cryptography and Network Security: Principles and Practice, 6th Edition* (6th ed., pp. 49–52). https://www.pearson.com/us/higher-education/product/Stallings-Cryptography-and-Network-Security-Principles-and-Practice-6th-Edition/9780133354690.html?tab=contents

Stallings, W. (2017). Elliptic Curve Cryptography. In T. Johnson (Ed.), *Cryptography and Network Security Principles and Practice* (7th Edition, pp. 330–334). Pearson Education.

Stehlé, D., & Steinfeld, R. (2011). Making NTRU as secure as worst-case problems over ideal lattices. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6632 LNCS*, 27–47. https://doi.org/10.1007/978-3-642-20465-4_4

Stillwell, J. (2003). Ideals. *Elements of Number Theory*, 196–220. https://doi.org/10.1007/978-0-387-21735-2_11

Strang, G. (2016). Cramer's Rule, Inverses, and Volumes. In *Introduction to Linear Algebra* (5th ed., pp. 417–422). Wellesley-Cambridge Press.

Sultan, S. (2019). Privacy-Preserving Metering in Smart Grid for Billing, Operational Metering, and Incentive-Based Schemes: A Survey. *Computers and Security*, *84*, 148–165. https://doi.org/10.1016/j.cose.2019.03.014

Sun, X., Wang, T., Sun, Z., Wang, P., Yu, J., & Xie, W. (2017). An Efficient Quantum Somewhat Homomorphic Symmetric Searchable Encryption. *International Journal of Theoretical Physics 2017 56:4*, *56*(4), 1335–1345. https://doi.org/10.1007/S10773-017-3275-0

Takayasu, A., & Kunihiro, N. (2019). Partial Key Exposure Attacks on RSA: Achieving the Boneh–Durfee Bound. *Theoretical Computer Science*, *761*, 51–77. https://doi.org/10.1016/j.tcs.2018.08.021

Takayasu, A., & Kunihiro, N. (2014). Partial Key Exposure Attacks on RSA: Achieving the Boneh-Durfee Bound. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8781*, 345–362. https://doi.org/10.1007/978-3-319-13051-4_21

Texas Instruments Incorporated. (2018). *MSP430FR596x, MSP430FR594x Mixed-Signal Microcontrollers*. https://www.ti.com/lit/ds/symlink/msp430fr5969.pdf

Thakur, K., & P., B. (2016). BTRU, A Rational Polynomial Analogue of NTRU Cryptosystem. *International Journal of Computer Applications*, *145*(12), 22–24. https://doi.org/10.5120/ijca2016910769

Thakur, K., & Tripathi, B. P. (2017). STRU: A Non Alternative and Multidimensional Public Key Cryptosystem. In *Global Journal of Pure and Applied Mathematics* (Vol. 13, Issue 5). http://www.ripublication.com/gjpam.htm

The LLL Algorithm - Survey and Applications | Phong Q. Nguyen | Springer. (n.d.). Retrieved May 31, 2021, from https://www.springer.com/gp/book/9783642022944

van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully Homomorphic Encryption over the Integers. *Lecture Notes in Computer Science*

(Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *6110*, 24–43. https://doi.org/10.1007/978-3-642-13190-5_2

Vats, N. (2009). NNRU, a noncommutative analogue of NTRU. *CoRR*, *abs/0902.1891*. http://arxiv.org/abs/0902.1891

Wang, B., Lei, H., & Hu, Y. (2018). D-NTRU: More efficient and average-case IND-CPA secure NTRU variant. *Information Sciences*, *438*, 15–31. https://doi.org/10.1016/j.ins.2018.01.037

Wang, L., & Ahmad, H. (2016). *Challenges of Fully Homomorphic Encryptions for the Internet of Things*. https://doi.org/10.1587/transinf.2015INI0003

Wang, X., Luo, T., & Li, J. (2018). *A More Efficient Fully Homomorphic Encryption Scheme Based on GSW and DM Schemes*. https://doi.org/10.1155/2018/8706940

Wiener, M. J. (1990). Cryptanalysis of Short RSA Secret Exponents. *IEEE Transactions on Information Theory*, *36*(3), 553–558. https://doi.org/10.1109/18.54902

Wu, M. E., Chen, C. M., Lin, Y. H., & Sun, H. M. (2014). On the Improvement of Wiener Attack on RSA with Small Private Exponent. *The Scientific World Journal*, *2014*. https://doi.org/10.1155/2014/650537

Yakubu, J., Abdulhamid, S. M., Christopher, H. A., Chiroma, H., & Abdullahi, M. (2019). Security challenges in fog-computing environment: a systematic

appraisal of current developments. In *Journal of Reliable Intelligent Environments* (Vol. 5, Issue 4, pp. 209–233). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/s40860-019-00081-2

Yang, H. M., Xia, Q., Wang, X. F., & Tang, D. H. (2012). A New Somewhat Homomorphic Encryption Scheme over Integers. *Proceedings - 2012 International Conference on Computer Distributed Control and Intelligent Environmental Monitoring, CDCIEM 2012*, 61–64. https://doi.org/10.1109/CDCIEM.2012.21

Yang, Z., Fu, S., Qu, L., & Li, C. (2018a). *Supplementary File to A Lower Dimension Lattice Attack on NTRU*. http://scis.scichina.com/en/2018/059101-supplementary.pdf

Yang, Z., Fu, S., Qu, L., & Li, C. (2018b). A Lower Dimension Lattice Attack on NTRU. In *Science China Information Sciences* (Vol. 61, Issue 5, pp. 1–3). Science in China Press. https://doi.org/10.1007/s11432-017-9175-y

Yassein, H. R., & Al-Saidi, N. M. (2016). HXDTRU Cryptosystem Based On Hexadecnion Algebra. *Proceeding of the 5th International Cryptology and Information Security Conference, Kota Kinabalu, Malaysia*.

Youn, T.-Y., Jho, N.-S., & Chang, K.-Y. (2016). Design of Additive Homomorphic Encryption with Multiple Message Spaces For Secure and Practical Storage Services over Encrypted Data. *The Journal of Supercomputing 2016 74:8, 74*(8), 3620–3638. https://doi.org/10.1007/S11227-016-1796-6

Yu, Y., Xu, G., & Wang, X. (2017). Provably Secure NTRU Instances over Prime Cyclotomic Rings. In S. Fehr (Ed.), *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I* (Vol. 10174, pp. 409–434). Springer. https://doi.org/10.1007/978-3-662-54365-8_17

Zhao, Y., Tarus, S. K., Yang, L. T., Sun, J., Ge, Y., & Wang, J. (2020). Privacy-Preserving Clustering for Big Data in Cyber-Physical-Social Systems: Survey and Perspectives. *Information Sciences*, *515*, 132–155. https://doi.org/10.1016/j.ins.2019.10.019

APPENDICES

Appendix A: Example of RSA Encryption/ Decryption

Example A.1: Example of 40-bit RSA encryption/decryption.

Let $p = 2^{20} + 33 = 1048609$ and $q = 2^{20} + 13 = 1048589$ be two prime numbers. Then modulus $N = p \cdot q = 1099559862701$. According to (2.3), let encryption exponent, $e = 2^{16} + 1 = 65537$. According to (2.5), decryption exponent, $d = 1082377437569$. The public key is $(N, e) = (1099559862701, 65537)$, and the private key is $(N, d) = (1099559862701, 1082377437569)$. Let the message, $m = 986648$, then the ciphertext is calculated according to (2.2):

$$c = m^e mod\ N = 480808351840. \qquad \text{(A.1)}$$

Message, m, is retrieved by decryption of the ciphertext (A.1) according to (2.4) as shown in (A.2):

$$m = c^d\ mod\ N = 986648. \qquad \text{(A.2)}$$

Appendix B: Lattice and Lattice Basis Reduction Algorithms

Lattices are defined according to Definition B.1 below

Definition B.1: (Hoffstein et al., 2014b, p. 388) Let $v_1, \dots, v_n \in \mathbb{R}^m$ be a set of linearly independent vectors. The lattice $\mathcal{L}$ generated by $v_1, \dots, v_n$ is the set of linear combinations of $v_1, \dots, v_n$ with coefficients in $\mathbb{Z}$,

$$\mathcal{L} = \{a_1 v_1 + a_2 v_2 + \cdots + a_n v_n : a_1, a_2, \dots, a_n \in \mathbb{Z}\}.$$

Definition B.2: (Hoffstein et al., 2014b, p. 388) A set of vectors $v_1, \dots, v_n \in R^m$ is (linearly) independent if the only way to get

$$a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = 0$$

is to have $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$.

The rank of the lattice $\mathcal{L}$ is n and its dimension is m, $n \leq m$. A full rank lattice is the lattice having its rank and dimensions are equal i.e., $n = m$. basis for $\mathcal{L}$ is any set of independent vectors $V = \{v_1, \dots, v_n\}$ that generates $\mathcal{L}$. The same lattice can be represented by different bases, any two matrices $B1$ and $B2$ associated with the same lattice, L, are related by an integer matrix U with $|det(U)| = 1$, that is, $B1 = U \times B2$. Therefore, $det(B1) = \pm\, det(B2)$. Determinant of lattice L is defined as follows

$$\det(L) = |\det(B)|, \qquad \text{(B.1)}$$

where B is a basis matrix of L. Figure B.1 illustrates two different bases for the same lattice. The first basis is "good" in the sense that the vectors are short, "nearly

orthogonal"; the second basis is "bad" because the vectors are long and quite skewed, i.e., the angle between the basis vectors is small.

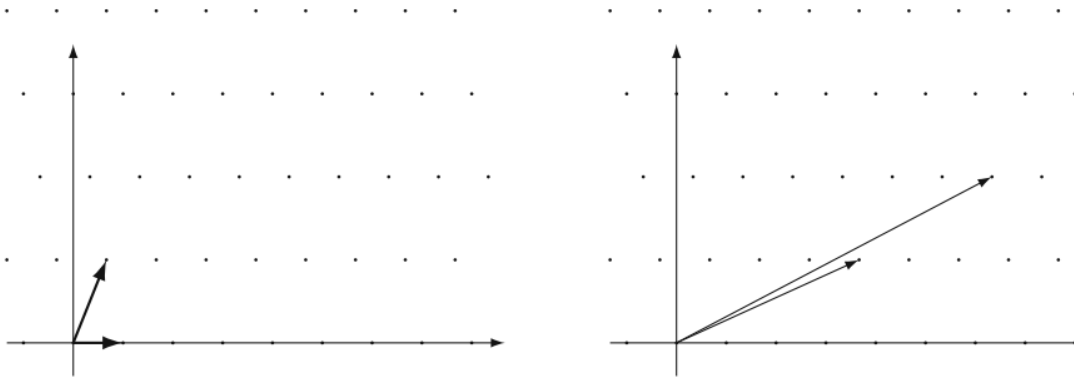

Figure B.1: Two different bases for the same lattice.
Note. Adapted from (Hoffstein et al., 2014b, p. 405).

Hadamard ratio can be used to check the orthogonality (wellness) of basis, where Hadamard ratio is defined according to Definition B.3 below

Definition B.3: (Hoffstein et al., 2014b, p. 397) Let $\mathcal{L}$ be an n-dimensional lattice with the basis $\mathcal{B} = \{v_1, \dots, v_n\}$, then, the Hadamard ratio of the basis $\mathcal{B} = \{v_1, \dots, v_n\}$ is defined to be the quantity,

$$\mathcal{H}(\mathcal{B}) = \left(\frac{\det(\mathcal{L})}{\|v_1\|\|v_2\| \dots \|v_n\|}\right)^{1/n}, \tag{B.2}$$

where $0 < \mathcal{H}(\mathcal{B}) \leq 1$, and the closer that the value is to 1, the more orthogonal (good) the basis. A bad basis has the ratio close to 0.

The Hermite Normal Form (HNF) unique basis of $\mathcal{L}$ is bad basis H and defined according to Definition B.4 below

Definition B.4: (Joe, 2016, p. 439; Micciancio, 2001, p. 128) The basis H of $\mathcal{L}$ is in upper triangular HNF if it is represented by an $n \times n$ matrix with

1- $h_{ij} = 0$ for $i > j$
2- $0 \leq h_{ij} < h_{jj}$ for$1 \leq i < j \leq n$
3- $0 < h_{ii}$ for $0 \leq i \leq n$

that is the elements of the main diagonal are positive integers, and in each column, the elements above the diagonal are less than the diagonal and at least zero.

Example of HNF. Let L be a 3-dimensional lattice with the basis $v_1 = (-97,19,19)$, $v_2 = (-36,30,86)$, $v_3 = (-184, -64, 78)$. Thus, the basis matrix B is

$$B = \begin{bmatrix} -97 & 19 & 19 \\ -36 & 30 & 86 \\ -184 & -64 & 78 \end{bmatrix},$$

Determinant of lattice L associated with B is 859516 and computed using Maple according to (B.1),

$$v1 := Matrix([-97, 19, 19]):$$
$$v2 := Matrix([-36, 30, 86]):$$
$$v3 := Matrix([-184, -64, 78]):$$
$$L := Matrix([[v1], [v2], [v3]])$$

$$\begin{bmatrix} -97 & 19 & 19 \\ -36 & 30 & 86 \\ -184 & -64 & 78 \end{bmatrix}$$

$$detL := \mathrm{abs}(Determinant(L))$$

$$859516$$

Hadamard ratio is approximately 0.746 and computed using Maple according to (B.2) as follows:

$$evalf\left(\left(\frac{detL}{norm(v1,2)\cdot norm(v2,2)\cdot norm(v3,2)}\right)^{\frac{1}{3}}\right)$$
$$0.7462020373$$

Hence, the basis is rather good as its Hadamard ratio is close to 1. Let's compute the bad basis HNF of$L$ using Maple:

$$HNF\_L := HermiteForm(L)$$
$$\begin{bmatrix} 1 & 1 & 285737 \\ 0 & 2 & 181486 \\ 0 & 0 & 429758 \end{bmatrix}$$

Hadamard ratio of it is approximately 0.000338 and computed using Maple according to (B.2)

$$v11 := HNF\_L[1];$$
$$v12 := HNF\_L[2];$$
$$v13 := HNF\_L[3];$$
$$\begin{bmatrix} 1 & 1 & 285737 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 2 & 181486 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 429758 \end{bmatrix}$$
$$evalf\left(\left(\frac{detL}{norm(v11,2)\cdot norm(v12,2)\cdot norm(v13,2)}\right)^{\frac{1}{3}}\right)$$
$$0.0003378626069$$

Thus, we see that the basis B is better than the bad basis HNF of L, skewed and having large norm vectors

Let $R = \frac{\mathbb{Z}[x]}{x^n+1}$ be the ring of polynomials with integer coefficients modulo $x^n + 1$, and I be an ideal of R. Ideal I is a subset of R that is closed under addition and multiplication by elements of R (Gentry & Halevi, 2011, p. 138). Thus, I satisfies

1- $0 \in I \subseteq R$
2- If $f, g \in I$, then $f + g \in I$.
3- If $f \in I$ and $h \in R$, then $hf \in I$.

For example, I can be the set of polynomials with all with integer even coefficients of degree up to $n - 1$. The set $(v) = \{vr : r \in R\}$ of all multiples of any $v \in R$ is an ideal, called the principal ideal generated by v (Stillwell, 2003, p. 196). Each element of R is a polynomial of degree at most $n - 1$, and thus is associated to a coefficient vector in Z^n. This way, we can view each element of R as a polynomial and a vector. The Euclidean norm $\|x\|$ of vector $x = (x_1, \dots, x_p)$ is defined as $\left(\sum_{i=1}^{p} x_i^2\right)^{1/2}$ (J. Han et al., 2012, p. 78).

The ideal $(\vec{j})$ generated by $\vec{j} \in R$ corresponds to the lattice, J, generated by the basis vectors $\left\{\overrightarrow{j_i} = \vec{j} \cdot x^i \ mod\ (x^n + 1) : i \in [0, n-1]\right\}$. "Every Basis $B = (\vec{b}_0, \dots, \vec{b}_{n-1})$ has a corresponding half open parallelepiped $\mathcal{P}(B) = \sum_{i=0}^{n-1} z_i \overrightarrow{b_i} : z_i \in \left(-\frac{1}{2}, \frac{1}{2}\right]$(Martins et al., 2017, p. 83: 6)". The length of the shortest nonzero vector in a lattice L is denoted $\lambda_1(L)$.

Every lattice induces a congruence relation, wherein two vectors, $\vec{x}$, $\vec{y}$, are congruent if $\vec{x} - \vec{y} \in \mathcal{L}$. The reduction of a vector $\vec{y}$ modulo lattice basis B, $\vec{x} = \vec{y}\ mod\ B$, corresponds to determining $\vec{x} \in \mathcal{P}(B)$ congruent with $\vec{y}$. This operation can be computed as (Martins, Sousa, and Mariano 2017, 83: 6 − 83: 7)

$$\vec{y}\ mod\ B = \vec{y} - \lfloor \vec{y} \times B^{-1} \rceil \times B, \qquad \text{(B.3)}$$

where × is a vector-matrix multiplication, $\lfloor \cdot \rceil$ means rounding to the nearest integer.

Shortest vector problem (SVP) is one of the most widely studied computational problem on lattices (Micciancio, 2016) is defined as follows:

Definition B.5: (Chuang et al., 2018; Hoffstein et al., 2014b, p. 395) Given a linearly independent basis $V = \{v_1, \dots, v_n\} \in \mathbb{R}^{m \times n}$ that generates $\mathcal{L}$, find a nonzero vector v such that $\|v\| = \min_{x \in V} \|x\|$ i.e., $v \in L$ that minimizes the Euclidean norm $\|v\|$.

The Euclidean norm $\|x\|$ of vector $x = (x_1, \dots, x_p)$, defined as $\left(\sum_{i=1}^{p} x_i^2\right)^{1/2}$ (J. Han et al., 2012, p. 78).

Remark 1: There may be more than one solution to the SVP.

For example, the integer lattice $\mathbb{Z}^2$, is the set of all 2-dimensional vectors with integer entries. Integer lattice $\mathbb{Z}^2$ can be represented by basis vectors V_1 = (1,1) and V_2= (1, 2), while the four nonzero vectors (0, ±1), (±1,0) are the solutions to the SVP.

Minkowski's Second Theorem (*The LLL Algorithm - Survey and Applications | Phong Q. Nguyen | Springer*, n.d., p. 35), sets an upper bound for the norm, l, of the shortest nonzero vector in a full rank 2-dimensional lattice given by (B.4):

$$\lambda \leq \sqrt{\gamma_2}\, \det(L)^{1/2}, \qquad \text{(B.4)}$$

where $\gamma_2 = \frac{2}{\sqrt{3}} \approx 1.154$ is Hermit's constant (*The LLL Algorithm - Survey and Applications | Phong Q. Nguyen | Springer*, n.d., p. 41), λ is the norm of the shortest

nonzero vector, and $det\big(L(V_1, V_2)\big)$ is the determinant of the lattice matrix formed by its basis vectors. Hence,

$$\lambda \leq \sqrt{1.154 \det(L)} \approx 1.07 \sqrt{\det(L)}. \tag{B.5}$$

Gaussian lattice reduction algorithm (GLR) (Hoffstein et al., 2014c) proposed by Gauss in the 19th century, and shown in Algorithm B.1 (our Maple implementation is shown in Code B.1) solves SVP in a 2-dimensioal lattice.

GLR algorithm shown in Algorithm B.1, upon termination returns the shortest vector v_1 in L generated by the basis $V = \{V_1, V_2\}$:

Algorithm B.1: GLR algorithm pseudocode (Hoffstein et al., 2014b, p. 437).

input: basis vectors V_1, V_2
output: the shortest vector v_1 in L
begin
1. $v_1 = V_1$; $v_2 = V_2$;
2. Loop
3. If $||v_2|| < ||v_1||$
4. swap v_1 and v_2.
5. Compute $m = \left\lfloor v1 \cdot v2 / ||v1||^2 \right\rceil$
6. If $m = 0$
7. return the shortest vector v1 of the basis, $\{v_1, v_2\}$.
8. Replace v_2 with $v_2 - mv_1$.

End Loop

Code B.1: Maple implementation for GLR Algorithm in Algorithm B.1

```
1.  GLR := proc (f, g, h, q)
2.  local fg, fgnorm, v1norm, v2norm, a, tmp, i;
3.  global result, counter, flag, v1, v2;
4.  flag := 0;
5.  v1 := Vector[column]([1, h]);
6.  v2 := Vector[column]([0, q]);
7.  fg := Vector[column]([f, g]);
8.  result := Matrix(2, 1);
9.  fgnorm := evalf(norm(fg, 2));
10. a := 1;
11. counter := 0;
12.  while a <> 0 do
13.   v1norm := evalf(norm(v1, 2));
14.   v2norm := evalf(norm(v2, 2));
15.   if v2norm < v1norm then
16.      tmp := v1;
17.      v1  := v2;
18.      v2 := tmp
19.   end if;
20.   if f = v1[1] or f = v2[1] then
21.      if g = v1[2] or g = v2[2] then
22.          flag := 1;
23.          print("keys are found")
24.      end if
25.   end if;
26.   if v1norm <= 10*fgnorm then
27.      result := <result| v1>;
28.   end if;
29.   if v2norm <= 10*fgnorm then
30.      result := <result| v2>
31.   end if;
32.   a := round(DotProduct(v1, v2)/norm(v1, 2)^2);
33.   v2 := v2-a*v1;
34.   counter := counter+1
35.   end do;
36. print(flag);
37. result := abs(result);
38. result := DeleteColumn(result, [1])
39. end proc
```

However, Solving SVP for higher dimensional lattice remains unsolved until LLL (Lenstra et al., 1982) proposed in 1982. LLL, shown in Algorithm B.2, runs in a polynomial time to solve approximate shortest vector problem (apprSVP) for higher dimension lattices. apprSVP defined as follows

Algorithm B.2: The LLL lattice reduction algorithm (Hoffstein et al., 2014b, p. 444)

input: basis vectors $\{v_1, \dots, v_n\}$ for the lattice L
output: Reduced basis vector $\{v_1, \dots, v_n\}$
begin

1. Set $k = 2$
2. Set $v_1^* = v_1$
3. Loop while $k \leq n$
4. Loop Down $j = k-1, k-2, \dots, 2, 1$
5. Set $v_k = v_k = \lfloor \mu_{k,j} \rceil v_j$
6. End j Loop
7. If $\|v_k^*\|^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|v_{k-1}^*\|^2$
8. Set $k = k + 1$.
9. Else
10. Swap v_{k-1} and v_k
11. Set $k = \max(k-1, 2)$
12. End If
13. End k loop
14. Return LLL reduced basis $\{v_1, \dots, v_n\}$

End

Definition B.6: (Hoffstein et al., 2014b, p. 409) Let $\phi(n)$ be a function of n. In a lattice L of dimension n, the approximate shortest vector problem is to find a nonzero vector that is no more than $\phi(n)$ times longer than a shortest nonzero vector. In other words, if $v_{shortest}$ is a shortest nonzero vector in L, find a nonzero vector $v \in L$ satisfying

$$\|v\| \leq \phi(n)\|v_{shortest}\|.$$

LLL solves apprSVP to within a factor of 2^n , where n is the dimension of the lattice. On termination, LLL returns a set of short vectors, beginning with the shortest vector found, and then with vectors whose lengths increase as slowly as possible until we reach the last vector in the basis in the lattice. The shortest vector v_1 found in an n-dimension lattice L satisfies

$$\|v_1\| \leq 2^{(n-1)/2} \min_{0 \neq v \in L} \|v\| \tag{B.6}$$

Appendix C: Examples of CPKC Scheme Encryption/ Decryption

Example C.1: Example of CPKC encryption/ Decryption. The example is close to Example 7.1, from (Hoffstein et al., 2014b, p. 375).

Key creation: Let according to (2.29), (2.30), $q = 122430513839$, $f = 231233$, and $g = 195696$. According to (2.31), $F_g = 127505$, and $F_q = 54368439252$ as shown in (1) and (2), Figure C.1 Public key component, h, is calculated by (2.32) as shown in (3) of Figure C.1:

$$h = F_q \cdot g \; mod \; q = 107143708775.$$

Public key is (h, q), and private key is $(f\ , g)$.

Encryption: Let according to (2.33) and (2.34), $r = 10101$ and $m = 12345$. The ciphertext, e as showin in (4) of Figure C.1, is computed according to (2.35):

$$e = r \cdot h + m \, mod \; q = 95290525699.$$

In Step 1 of the decryption process, equation (2.36) is applied as shown in (5) of Figure C.1:

$$a = f \cdot e \, mod \; q = r \cdot g + f \cdot m = 4831296681.$$

In Figure C.1, the message m is retrieved using (2.39) as shown in (6), Figure S1:

$$m = F_g \cdot a \, mod \; g = 12345.$$

Thus, the plaintext $m = 12345$ is revealed. It can be seen that CPKC encryption/decryption procedure (2.35), (2.36), and (2.38), works correctly due to (2.37) holding.

$q := 122430513839$:
$f := 231233$:
$g := 195696$:

$igcdex(f, q, 'Fq', 'qz')$:
$igcdex(f, g, 'Fg', 'gz')$:

$Fq \bmod q$

$$54368439252 \quad (1)$$

$Fg \bmod g$

$$127505 \quad (2)$$

$h := Fq \cdot g \bmod q;$

$$107143708775 \quad (3)$$

$r := 10101$:
$m := 12345$:
$e := r \cdot h + m \bmod q$

$$95290525699 \quad (4)$$

$a := f \cdot e \bmod q$

$$4831296681 \quad (5)$$

$Fg \cdot a \bmod g$

$$12345 \quad (6)$$

$GLR(f, g, h, q)$:

"keys are found"
"keys are found" (7)

$v1$

$$\begin{bmatrix} 231233 \\ 195696 \end{bmatrix} \quad (8)$$

$counter$

$$9 \quad (9)$$

Figure C.1: Screenshot of LBRA by GLR using Maple code in Appendix B on CPKC for the data from the Example C.1 finding the private key components, $(f, g) = v1$, in 9 iterations.

Example C.2: Example of LBRA using GLR against CPKC

In this example, we try LBRA by GLR using Maple code in AppendixB on CPKC private key/message for the data from the Example C.1. LBRA by GLR finds in 9 iterations the shortest vector, $v_1 = (231233, 195696)$ as shown in Figure C.1. The shortest vector, v_1, found by GLR corresponds to the private key components, (f, g), because they were selected small, having order $\mathcal{O}(\sqrt{q})$ values according to (2.40). Note that the norm of the vector, $(f, g) = \sqrt{f^2 + g^2} = 3.029284141 \times 10^5$ is small compared to $\sqrt{q} = 3.499007199 \times 10^5$. The message-related vector, $(r, e - m)$, is not disclosed in the attack because $e = \mathcal{O}(q)$.

Appendix D: Example of HE1N Encryption/Decryption

Example D.1: Example of HE1N encryption/decryption.

Let $\lambda = 14$, $\rho = 8$, $\rho' = 11$ be the inputs of Algorithm 2.2, and let the parameters (p, v, k) selected by Algorithm 2.2 to be $p = 8200 \in [2^{13}, 2^{14}]$, $v = 3$, and $k = 8 \in [2^2, 2^3]$. Algorithm 2.2 outputs the key (k, p).

Next, Algorithm 2.3 inputs (λ, ρ', k, p). Algorithm 2.3 assigns $\eta \leftarrow 6 \approx \lambda^2/\rho' - \lambda = 5.8181$. Let $q = 32 \in [2^5, 2^6]$. Algorithm 2.3 assigns modulus $\leftarrow 262400 = \text{p} \cdot \text{q}$.

Let the input plaintext messages bounded by $M = 32$. Encryption of $m = 7 \in [0,32)$ is performed by Algorithm 2.4. Let (r, s) selected by Algorithm 2.4 to be $r = 15 \in [1, q)$, $s = 2 \in [0, k)$. Algorithm 2.4 finds c, the ciphertext encrypts m, $c = m + sk + rp \; mod$ modulus $= 7 + 2 \cdot 8 + 15 \cdot 8200 \bmod 262400 =$ $123023 \bmod 262400 = 123023$. **Note that modulus operation in the encryption step, has no effect, as we proved in Section 5.1.**

The message m is retrieved by Algorithm 2.5. $m = (c \; mod \; p) \; mod \; k = (12302323 \; mod \; 8200) \; mod \; 8 = 7$

Appendix E: Examples of RLWE-NCM-CSCM Cryptosystem Encryption/ Decryption

Example E.1: Example of failing decryption when condition (2.56) is not satisfied.

Let $n = 4$, $R = \mathbb{Z}[x]/(x^4 + 1)$, $q = 7$,$t = 2$, $m = x + 1$, $e = x^3 + 2x^2$, $a = x^2$, $s = x^2 - 3$. Note that $|2e + m| = 2x^3 + 4x^2 + x + 1$, thus condition (2.56) is violated since $4 > 3.5 = q/2$. Encryption is performed according to (2.53), $c = \binom{c_0}{c_1} = \binom{2x^3 + x^2 + x}{-x^2} \in R_q^2$. Decryption is performed according to (2.54), $c_0 + c_1 \cdot s \; mod \; (x^4 + 1) \; mod \; 7 \; mod \; 2 = 2x^3 - 3x^2 + x + 1 \; mod \; 2 = x^2 + x + 1 \neq m$.

Example E.2: Example of successful decryption when condition (2.56) is satisfied.

Let $n = 4$, $R = \frac{\mathbb{Z}[x]}{x^4+1}$, $q = 7$,$t = 2$, $m = x + 1$, $e = x^3 + x^2$,$a = x^2$, $s = x^2 - 3$. Note that $2e + m = 2x^3 + 2x^2 + x + 1$, thus condition (2.56) is satisfied as $q/2 = 3.5$. Encryption is performed according to (2.53), $c = \binom{c_0}{c_1} = \binom{2x^3 - x^2 + x}{-x^2} \in R_q^2$. Decryption is performed according to (2.54), $c_0 + c_1 \cdot s \; mod \; (x^4 + 1) \; mod \; 7 \; mod \; 2 = 2x^3 + 2x^2 + x + 1 \; mod \; 7 \; mod \; 2 = x + 1 = m$.

Appendix F: Example of RCPKC Scheme Encryption/ Decryption

Example F.1 aims to show the process of finding RCPKC random interval, and how LBRA using GLR fails to compromise RCPKC private key/message. For calculations, Maple is used. See Figure F.1.

Example F.1: Example of Finding RCPKC Random Interval, and LBRA by GLR Failure

Key Creation: Let $mgLen = 16$, $qLen = 80$, meeting (6.10), $q = 2^{qLen}$, private key components, $g = 65{,}535$, and $f = 1,351,417,702,001$, are selected to meet (6.6) and (6.8) respectively as shown in (2) and (3) of Figure F.1. Values F_q and F_g are found in (4) and (5) of Figure F.1. The public key component, h, is calculated according to (2.32) as shown in (6) of Figure S4.

Finding Random Interval: To select random $\boldsymbol{r}$, GLR algorithm shown in Appendix B is launched with inputs $V1 = (1, h)$ and $V2 = (0, q)$. GLR terminates in 18 iterations as shown in (8) of Figure F.1, with 5 pairs (F_i, G_i) satisfying (6.12) shown in (7) of Figure F.1, it is noticed that none of these vectors is equal to (f, g). Hence, (6.13) is satisfied. Maximum F_i and minimum G_i are found in (9), and (10) of Figure F.1; value $rmin$ is defined according to (6.14) as shown in (11), $rmin$ also satisfies (6.15) as shown in (18) of Figure F.1. $rmax$ is calculated in (12) of Figure F.1. After calculating $max(\alpha \cdot 2^{qlen/2}, rmin)$ in (13) of Figure F.1, it is perceived that (6.19) is satisfied as shown in (14) of Figure F.1. Thus, r is selected from (6.18) as shown in (15) of Figure F.1.

LBRA using GLR Failure: For the message $m = 14$, it is noticed that decryption correctness condition (6.17) is valid using private key (f, g) as shown in (16) of Figure F.1, and not valid for (F_i, G_i) returned by GLR as shown in (17) of Figure F.1. Hence, GLR attack fails to return keys usable for ciphertext decryption.

$mgLen := 16$:
$qLen := 80$:
$q := 2^{qLen}$:
$f := 1351417702001$:
$g := 2^{16} - 1$:
$gcd(f, q \cdot g)$

$$1 \tag{1}$$

$is(g < 2^{mgLen}$ **and** $g \geq 2^{mgLen-1})$

$$true \tag{2}$$

alpha $:= 1.07$:
$is\left(f < 2^{qLen - mgLen - 1}\right.$ **and** $\left. f \geq \text{alpha} \cdot 2^{\frac{qLen}{2}}\right)$

$$true \tag{3}$$

$igcdex(f, q, 'Fq', 'qz')$:
$igcdex(f, g, 'Fg', 'gz')$:
Fq

$$154260404770580979079825 \tag{4}$$

Fg

$$2291 \tag{5}$$

$h := Fq \cdot g$ **mod** q;

$$41792302249530510328766 3 \tag{6}$$

$GLR(f, g, h, q)$

$$0$$

$$\begin{bmatrix} 1133958117 & 459459339518 & 26960316553 & 459459339518 & 891958362483 \\ 44728059201205 & 894561206306 & 2683683553383 & 894561206306 & 894561140771 \end{bmatrix} \tag{7}$$

$counter$

$$18 \tag{8}$$

$maxF := \max(result[1, 1..ColumnDimension(result)])$

$$891958362483 \tag{9}$$

$minG := \min(result[2, 1..ColumnDimension(result)])$

$$894561140771 \tag{10}$$

$rmin := \text{ceil}\left(\frac{q + g \cdot maxF}{minG}\right)$

$$1351417832690 \tag{11}$$

$rmax := \text{floor}\left(\frac{q}{g} - f\right)$

$$18447024201563593104 \tag{12}$$

$\max\left(\text{trunc}\left(\text{alpha} \cdot 2^{\frac{qLen}{2}}\right), rmin\right)$

$$1351417832690 \tag{13}$$

$is\left(rmax > 2 \cdot \max\left(\text{trunc}\left(\text{alpha} \cdot 2^{\frac{qLen}{2}}\right), rmin\right)\right)$

$$true \tag{14}$$

$r := rmin + 1024;$

$$1351417833714 \tag{15}$$

$m := 14$:
$is(|r \cdot g + f \cdot m| < q)$

$$true \tag{16}$$

$F1 := result[1, 1] : F2 := result[1, 2] : F3 := result[1, 3] : F4 := result[1, 4] : F5 := result[1, 5]$:
$G1 := result[2, 1] : G2 := result[2, 2] : G3 := result[2, 3] : G4 := result[2, 4] : G5 := result[2, 5]$:

$is(|r \cdot G1 + F1 \cdot m| < q$ **or** $|r \cdot G2 + F2 \cdot m| < q$ **or** $|r \cdot G3 + F3 \cdot m| < q$ **or** $|r \cdot G4 + F4 \cdot m| < q$ **or** $|r \cdot G5 + F5 \cdot m| < q)$

$$false \tag{17}$$

$is(h \cdot rmin > q)$

$$true \tag{18}$$

Figure F.1. Screenshot of Maple code of Example F.1.

Appendix G: NTRU Asymmetric Encryption Padding IND-CCA2 Security (NAEP)

NTRU asymmetric encryption padding (NAEP) (Howgrave-Graham, Silverman, Singer, et al., 2003) has been proven IND-CCA2 secure. In the following, NAEP is introduced, and then, its IND-CCA2 security is discussed.

NAEP Description: NAEP uses a function,

$$compress(p(x)) = p(x)\ mod\ q\ mod\ 2, \qquad \text{(G.1)}$$

where $p(x)$ is a polynomial. NAEP encryption is introduced in Algorithm G.1.

Algorithm G.1: NAEP encryption

input: $N = \theta(k); N > l = \theta(k)$ is the padding size; $G: \{0,1\}^{N-l} \times \{0,1\}^{l} \rightarrow D_r$ and $H: \{0,1\}^{N} \rightarrow \{0,1\}^{N}$ are hash functions; $m \in \{0,1\}^{N-l}$ is the input plaintext message; h is the public key; q is the modulus value.

output: $e \in R_q$ is the ciphertext.

begin

1. Pick $\mu \leftarrow_{\$} \{0,1\}^{l}$, where $\leftarrow_{\$}$ means uniform random sampling
2. Let $\rho = G(m, \mu), r = genr(\rho), s = compress(p \cdot r \cdot h)$, and $\omega = (m||\mu) \oplus H(s)$. $genr$ is a function generating correct r; $\oplus$ denotes XOR;
3. If $\omega \notin \tilde{R}$, goto 1. $\tilde{R}$ is the space of binary polynomials with the number of ones such that the probability of NTRU decryption failure is negligible.
4. $e = F_{h,p}^{NTRU}(\omega, r)$, according to (2.20)

end

The $compress()$ binary string result is sed in Step 2 of NAEP encryption to hide the padded message by XOR operation and hashing. NAEP decryption is introduced in

Algorithm G.2: NAEP decryption

input: $N = \theta(k)$; $N > l = \theta(k)$ is the padding size; $G: \{0,1\}^{N-l} \times \{0,1\}^{l} \rightarrow D_r$ and $H: \{0,1\}^{N} \rightarrow \{0,1\}^{N}$ are hash functions; $e \in R_q$ is the ciphertext.

output: $m \in \{0,1\}^{N-l}$ is the decrypted plaintext message if decrypted correctly, and Reject otherwise.

begin

1. $a = center(f \cdot e \; mod \; q)$
2. $\omega = F_p \cdot a \; mod \; p$. According to NTRU step 2
3. $s = compress(e - w)$
4. $m||\mu = \omega \oplus H(s); r = genr\big(G(m||\mu)\big)$.
5. If $p \cdot r \cdot h = s \; mod \; q$, then output m; else, output Reject.

end

Security of NAEP IND-CCA2: NAEP has been proven to be IND-CCA2 secure (Howgrave-Graham, Silverman, Singer, et al., 2003).

Definition G.1: (Howgrave-Graham, Silverman, Singer, et al., 2003, p. 3) A time τ algorithm $\mathcal{A}$ is a $(\tau;\ \epsilon)$-chosen ciphertext algorithm, with advantage ϵ in attacking a randomized encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ if there is a pair of sub-algorithms

$$\mathcal{A}_1: PK \rightarrow \mathcal{M} \times \mathcal{M} \times \mathcal{S},$$

$$\mathcal{A}_2: \mathcal{C} \times \mathcal{S} \rightarrow \{0,1\},$$

such that if $(M_0, M_1, s) \leftarrow \mathcal{A}_1(pk)$ then

$$\left|\Pr[\mathcal{A}_2(c^*, s) = b^*] - \frac{1}{2}\right| = (1/2)\epsilon$$

where $c^* \leftarrow \mathcal{E}(M^*, r^*)$ for some $r^* \in \mathcal{R}_{\mathcal{E}}$, and $M^* = M_{b^*}$ for some $b^* \in \{0, 1\}$. This probability is defined over the choice of $r^* \leftarrow_{\$} \mathcal{R}_{\mathcal{E}}, b^* \in \{0, 1\}$ and $k \in \mathcal{R}_{\mathcal{K}}$, where $\mathcal{R}_{\mathcal{E}}$ and $\mathcal{R}_{\mathcal{K}}$ are defined below. The algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ have access to a decryption

oracle $\mathcal{D}$, which they can call on all but the challenge ciphertext c^*, but they must make all hash function calls to $H_1, \ldots H_n$ public.

An encryption scheme is IND-CCA2 secure if there exist no polynomial (on security parameter) time adversary with a non-negligible advantage. Key generation, encryption, and decryption algorithms are formalized as follows (Howgrave-Graham, Silverman, Singer, et al., 2003). For a given parameter set $\mathcal{P}$, the encryption scheme is specified by three algorithms:

$$\mathcal{K}: \mathcal{R}_{\mathcal{K}} \rightarrow \mathcal{PK} \times \mathcal{SK}$$

$$\mathcal{E}: \mathcal{PK} \times \mathcal{M} \times \mathcal{R}_{\mathcal{E}} \rightarrow \mathcal{C}$$

$$\mathcal{D}: \mathcal{SK} \times \mathcal{C} \rightarrow \mathcal{M},$$

called the key generation, encryption, and decryption algorithms, respectively. The spaces $\mathcal{R}_{\mathcal{K}}$, $\mathcal{PK}$, $\mathcal{SK}$, $\mathcal{M}$, $\mathcal{R}_{\mathcal{E}}$, $\mathcal{C}$ are called the key-gen randomness, public key, secret key, message, encryption randomness, and ciphertext space, respectively

If $(pk_k , sk_k) \leftarrow \mathcal{K}(k)$, then the algorithms should satisfy:

$$\mathcal{D}\big(sk_k, \mathcal{E}(pk_k, M, r)\big) = M$$

for all $k \in \mathcal{R}_{\mathcal{K}}, M \in \mathcal{M}$ and $r \in \mathcal{R}_{\mathcal{E}}$. NTRU key, encryption, and decryption procedures and respective spaces are defined in Section 2.3.1 according to (Howgrave-Graham, Silverman, Singer, et al., 2003). Polynomials used in NAEP (Howgrave-Graham et al., 2005; Howgrave-Graham, Silverman, Singer, et al., 2003) for keys are invertible. The NTRU one-way (NTRU-OW) problem is defined as follows:

Definition G.28: NTRU-OW problem: For a parameter set, $\mathcal{P}_{NTRU}$, we denote by $Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU})$ the success probability of a probabilistic polynomial time (PPT) adversary, $\mathcal{A}$, for finding a pre-image of $F_{h,p}^{NTRU}$,

$$Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU}) = \Pr\begin{pmatrix} (M', r') \leftarrow \mathcal{A}(e, h) \\ such\ that\ F_{h,p}^{NTRU}(M', r') = e \end{pmatrix}.$$

Assumption G.1: NTRU-OW assumption: For every PPT adversary, $\mathcal{A}$, solving the NTRU-OW problem, there exists a negligible function, $v_A(k)$, such that for sufficiently large k, it holds:

$$Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU}) \leq v_A(k).$$

Definition G.3: (Howgrave-Graham, Silverman, Singer, et al., 2003) A function $\nu: \mathbb{N} \rightarrow \mathbb{R}$ is said to be negligible if for every constant $c \geq 0$, there exists an integer k_c such that $\nu(k) < k^{-c}$ for all $k \geq k_c$.

NTRU variants (Hoffstein et al., 1999; Howgrave-Graham, Silverman, Singer, et al., 2003) can fail; hence, it was assumed in (Howgrave-Graham, Silverman, Singer, et al., 2003) that the failure probability is negligible. Under these assumptions, the IND-CCA2 security of NAEP was proven in (Howgrave-Graham, Silverman, Singer, et al., 2003), Corollary 1.

Appendix H: Formulas for CPU Power Consumption Calculation

Power, P, and energy, E, are measured in watts (W) and joules (J) (Smith, 2002, p. 1028), respectively, and calculated as follows:

$$P = V \cdot I, \tag{H.1}$$

$$E = P \cdot T, \tag{H.2}$$

where V is the potential difference measured in volts (V), I is the electric current measured in amperes (A), and T is the running time in seconds. There are three contributors to the CPU power consumption: dynamic, short-circuit, and power loss due to transistor leakage currents (Beloglazov et al., 2011):

$$P_{cpu} = P_{dyn} + P_{sc} + P_{leak}. \tag{H.3}$$

Power consumption is mainly defined by the dynamic and leakage components (Nikolić, 2008). Leakage power, caused by leakage currents, is present in any active circuit independent of clock rates and is calculated as follows:

$$P_{leak} = V \cdot I_{leak}, \tag{H.4}$$

where V is the supply voltage and I_{leak} is the leakage current. Dynamic power consumption depends on circuit activity (i.e., transistor switches, changes of values in registers, etc.) and is defined as follows:

$$P_{dyn} = a \cdot C \cdot V^2 \cdot f\,, \tag{H.5}$$

where a is the switching activity factor, C is the capacitance measured in farad (F), and f is the clock frequency measured in hertz (Hz). Mostly, the activity factor is $a = 0.5$ (Romli et al., 2015). MSP430FR5969, a Texas Instruments microcontroller with capacitance $C = 20\,pF$ (Texas Instruments Incorporated, 2018) (Table 5–12), active supply voltage from $1.8, \ldots, 3.6\,V$ (Texas Instruments Incorporated, 2018, p. 1), clock frequency from $1, \ldots, 16\,MHz$ (Texas Instruments Incorporated, 2018, p. 19), $I_{leak} = 20\,nA$ (Texas Instruments Incorporated, 2018) (Table 5–11), is used for RCPKC power consumption evaluation in Subsection 6.5.2

Appendix I: Examples of RLWE-CSCM

Example I.1: Example of correct encryption/ decryption when (7.1) holds:

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys' components, A, s, and e are defined as follows

$$A = 93x^2 + 110x + 17 = (x + 1)(93x + 17), \tag{I.1}$$

$$s = 78x^2 + 85x + 101, \tag{I.2}$$

$$e = 91x^2 + 58x + 62\ , \tag{I.3}$$

where A meets (7.2). From (I.1)-(I.3), and (7.3), the public key, $pk \in R_q^2$, is defined as follows:

$$pk = \begin{pmatrix} pk_1 \\ pk_2 \end{pmatrix} = \begin{pmatrix} A \cdot s + p \cdot e \\ -A \end{pmatrix} = \begin{pmatrix} 21x^2 + 37x + 51 \\ 12x^2 + 100x + 88 \end{pmatrix}. \tag{I.4}$$

From (I.2) and (7.4), it follows that secret key, $sk \in R_q^2$, is

$$sk = \begin{pmatrix} 1 \\ s \end{pmatrix} = \begin{pmatrix} 1 \\ 78x^2 + 85x + 101 \end{pmatrix}. \tag{I.5}$$

Encryption of the message,

$$m = 4x^2 + 5x + 1 \in R_p, \tag{I.6}$$

is performed using random r meeting (7.7),

$$r = 67x^2 + 49x + 14 \in R_q. \quad \text{(I.7)}$$

Ciphertext is computed according to (7.5),

$$c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r + m \\ pk_2 \cdot r \end{pmatrix} = \begin{pmatrix} 68x^2 + 40x + 42 \\ 44x^2 + 48x + 34 \end{pmatrix} \in R_q^2, \quad \text{(I.8)}$$

meeting (7.6). Decryption is executed through two steps in (7.8).

Step 1:

$$[\langle c, sk \rangle]_q = c_1 + c_2 \cdot s \ \ mod\ q = 39x^2 \ + \ 26x \ + \ 57, \quad \text{(I.9)}$$

with quotient $k = 551x^2 - 150x - 499$ in (7.10), found as follows:

$$\langle c, sk \rangle = p \cdot e \cdot r + m = 57894x^2 - 15724x - 52338 \in R$$

$$k = \left\lfloor \frac{\langle c,sk \rangle}{q} \right\rceil = \left\lfloor \frac{57894}{q} \right\rceil x^2 - \left\lfloor \frac{15724}{q} \right\rceil x - \left\lfloor \frac{52338}{q} \right\rceil = 551x^2 \ - \ 150x - 499.$$

Step2: Modulo p operation is applied to (I.9),

$$m' = \left[[\langle c, sk \rangle]_q \right]_p = 39x^2 \ + \ 26x \ + \ 57 \ mod\ 7 = 4x^2 + 5x + 1 = m, \quad \text{(I.10)}$$

Figure I.1 shows a screenshot of Maple code of Example I.1.

Example I.2: Example of correct encryption/ decryption when (7.1) does not hold:

Let $p = 7$, $q = 15 * \ p + t = 106, t = 1$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys' components, A, s, and e, are defined in (I.1)-(I.3). From (I.1)-(I.3) and (7.3), the public key $pk \in R_q^2$ is defined as follows:

Parameter Setup

$p := 7 :$

$q := 15 \cdot p :$

$d := 3 :$

$Z := x^d + 1 :$

Gernerating Public key and Secret key

$A := 93\,x^2 + 110\,x + 17 :$

$s := 78\,x^2 + 85\,x + 101 :$

$e := 91\,x^2 + 58\,x + 62 :$

$pk := Vector([rem(A \cdot s, Z, x) + p \cdot e, -A] \bmod q);$

$sk := Vector([1, s])$

$$\begin{bmatrix} 21\,x^2 + 37\,x + 51 \\ 12\,x^2 + 100\,x + 88 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 78\,x^2 + 85\,x + 101 \end{bmatrix}$$

Encrypting message, m

$m := 4\,x^2 + 5\,x + 1 :$

$r := 67\,x^2 + 49\,x + 14 :$

$c := Vector(2) :$

$c[1] := rem(pk[1] * r, Z, x) + m \bmod q :$

$c[2] := rem(pk[2] * r, Z, x) \bmod q :$

$c;$

$$\begin{bmatrix} 68\,x^2 + 40\,x + 42 \\ 44\,x^2 + 78\,x + 34 \end{bmatrix}$$

Decrypting ciphertext c1

$step1 := rem(expand(Multiply(Transpose(c), sk)), Z, x) \bmod q ;$

$step2 := step1 \bmod p$

$$39\,x^2 + 26\,x + 57$$

$$4\,x^2 + 5\,x + 1$$

find value of k.

$temp := rem(expand(p \cdot e \cdot r + m), Z, x) :$

$$k := \frac{(temp - (temp \bmod q))}{q}$$

$$551\,x^2 - 150\,x - 499$$

Figure I.1: Screenshot of Maple code of RLWE-CSCM encryption and success decryption when (7.1) holds using parameter settings of Example I.1

$$pk = \binom{pk_1}{pk_2} = \binom{A \cdot s + p \cdot e}{-A} = \begin{pmatrix} 36x^2 + 89x + 82 \\ 13x^2 + 102x + 89 \end{pmatrix}. \quad \text{(I.11)}$$

From (I.2) and (7.4) it follows that secret key, $sk \in R_q^2$, is

$$sk = \binom{1}{s} = \begin{pmatrix} 1 \\ 78x^2 + 85x + 101 \end{pmatrix}. \quad \text{(I.12)}$$

Encryption of the message in (I.6), using the random, r, in (I.7) is performed according to (I.5),

$$c = \binom{c_1}{c_2} = \binom{pk_1 \cdot r + m}{pk_2 \cdot r} = \begin{pmatrix} 81x^2 + 101x + 100 \\ 13x^2 + 42x + 29 \end{pmatrix} \in R_q^2. \quad \text{(I.13)}$$

Decryption is executed through two steps in (7.8).

Step 1:

$$[\langle c, sk \rangle]_q = c_1 + c_2 \cdot s \ mod\ q = 18x^2 + 70x + 26. \quad \text{(I.14)}$$

Step2: Modulo p operation is applied to (I.14) , and, recalling (I.6),

$$m' = \left[[\langle c, sk \rangle]_q\right]_p = 18x^2 + 70x + 26\ mod\ 7 = 4x^2 + 5 \neq m = 4x^2 + 5x + 1,$$

where quotient $k = 546x^2 - 148x - 493$ in (7.13), found as follows:

$$\begin{aligned} \langle c, sk \rangle &= p \cdot e \cdot r + m = 57894x^2 - 15724x - 52338 \in R \\ k &= \lfloor \langle c, sk \rangle / q \rceil = \lfloor 57894/q \rceil x^2 - \lfloor 15724/q \rceil x - \lfloor 52338/q \rceil \\ &= 546x^2 - 149x - 494. \end{aligned}$$

Thus, decryption failed as by (7.14) $m' = 4x^2 + 5 = m - kt\ mod\ p = 4x^2 + 5x + 1 - 5x - 3\ mod\ 7 = 4x^2 + 5 \neq m$. Figure I.2 shows a screenshot of Maple code of Example I.2.

Parameter Setup

$p := 7 :$

$q := 15 \cdot p + 1 :$

$d := 3 :$

$Z := x^d + 1 :$

Gernerating Public key and Secret key

$Ap := 93\,x^2 + 110\,x + 17 :$

$sp := 78\,x^2 + 85\,x + 101 :$

$e := 91\,x^2 + 58\,x + 62 :$

$pk := Vector([rem(Ap \cdot sp, Z, x) + p \cdot e, -Ap] \bmod q);$

$s := 78\,x^2 + 85\,x + 101 :$

$sk := Vector([1, s])$

$$pk := \begin{bmatrix} 36\,x^2 + 89\,x + 82 \\ 13\,x^2 + 102\,x + 89 \end{bmatrix}$$

$$sk := \begin{bmatrix} 1 \\ 78\,x^2 + 85\,x + 101 \end{bmatrix}$$

Encrypting message, m

$m := 4\,x^2 + 5\,x + 1 :$

$r := 67\,x^2 + 49\,x + 14 :$

$c := Vector(2) :$

$c[1] := rem(pk[1] * r, Z, x) + m \bmod q :$

$c[2] := rem(pk[2] * r, Z, x) \bmod q :$

$c;$

$$\begin{bmatrix} 81\,x^2 + 101\,x + 100 \\ 13\,x^2 + 42\,x + 29 \end{bmatrix}$$

Decrypting ciphertext c1

$step1 := rem(expand(Multiply(Transpose(c), sk)), Z, x) \bmod q ;$

$step2 := step1 \bmod p$

$$step1 := 18\,x^2 + 70\,x + 26$$

$$step2 := 4\,x^2 + 5$$

find value of k.

$temp1 := rem(expand(p \cdot e \cdot r + m), Z, x) :$

$$k := \frac{(temp1 - (temp1 \bmod q))}{q}$$

$$k := 546\,x^2 - 149\,x - 494$$

Figure I.2: Screenshot of Maple code of RLWE-CSCM encryption and success decryption when (7.1) does not hold using parameter settings of Example I.2

Example I.3: Example of homomorphic addition of two terms

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys, A, s, and e are defined in (I.1)-(I.3). From (I.1)-(I.3), and (7.3), the public key $pk \in R_q^2$, and secret key, $sk \in R_q^2$ are defined in (I.4) and (I.5). respectively.

Encryption of the messages,

$$m^{(1)} = 4x^2 + 5x + 1 \in R_p, \tag{I.15}$$

$$m^{(2)} = 5x^2 + 2x + 4 \in R_p, \tag{I.16}$$

with $m^{(1)} + m^{(2)} = 2x^2 + 5$. Using the random polynomials,

$$r^{(1)} = 62x^2 + 71x + 58 \in R_q, \tag{I.17}$$

$$r^{(2)} = 67x^2 + 49x + 14 \in R_q, \tag{I.18}$$

is performed according to (7.5),

$$c^{(1)} = \begin{pmatrix} c_1^{(1)} \\ c_2^{(1)} \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r^{(1)} + m^{(1)} \\ pk_2 \cdot r^{(1)} \end{pmatrix} = \begin{pmatrix} 2x^2 + 93x + 56 \\ 27x^2 + 23x + 101 \end{pmatrix} \in R_q^2, \tag{I.19}$$

$$c^{(2)} = \begin{pmatrix} c_1^{(2)} \\ c_2^{(2)} \end{pmatrix} = \begin{pmatrix} pk_1 \cdot r^{(2)} + m^{(2)} \\ pk_2 \cdot r^{(2)} \end{pmatrix} = \begin{pmatrix} 69x^2 + 37x + 45 \\ 44x^2 + 78x + 34 \end{pmatrix} \in R_q^2. \tag{I.20}$$

Let

$$C = c^{(1)} + c^{(2)} = \begin{pmatrix} 71x^2 + 25x + 101 \\ 71x^2 + 101x + 30 \end{pmatrix} \in R_q^2.$$

Decryption is executed through two steps in (7.8).

Step 1:

$$[\langle C, sk \rangle]_q = C_1 + C_2 \cdot s mod\ q = 2x^2 + 98x + 33. \qquad \text{(I.21)}$$

Step 2: modulo p operation is applied to (7.21),

$$m' = \left[[\langle C, sk \rangle]_q\right]_p = 2x^2 + 98x + 33\ mod\ 7 = 2x^2 + 5 = m^{(1)} + m^{(2)} \in R_p.$$

Figure I.3 shows a screenshot of Maple code of Example I.3.

Example I.4: Example of homomorphism for $1000 \cdot c^{(1)} + 2000 \cdot c^{(2)}$

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys, A, s, and e are defined in (I.1)-(I.3). From (I.1)-(I.3), and (7.3), the public key $pk \in R_q^2$, and secret key, $sk \in R_q^2$ are defined in (I.4) and (I.5). respectively.

Encryption of the messages $m^{(1)}$ and $m^{(2)}$ in (I.15) and (I.16), using the random, $r^{(1)}$ and $r^{(2)}$ in (I.17) and (I.18) is $c^{(1)}$ and $c^{(2)}$ in (I.19) and (I.20). Let us calculate

$$1000 \cdot m^{(1)} + 2000 \cdot m^{(2)} = 5x + 5 \in R_p, \qquad \text{(I.22)}$$

Let

$$C = 1000 \cdot c^{(1)} + 2000 \cdot c^{(2)} = \begin{pmatrix} 35x^2 + 50x + 50 \\ 25x^2 + 80x + 55 \end{pmatrix} \in R_q^2, \qquad \text{(I.23)}$$

Parameter Setup

$p := 7$:

$q := 15 \cdot p$:

$d := 3$:

$Z := x^d + 1$:

Gernerating Public key and Secret key

$A := 93\,x^2 + 110\,x + 17$:

$s := 78\,x^2 + 85\,x + 101$:

$e := 91\,x^2 + 58\,x + 62$:

$pk := Vector([rem(A \cdot s, Z, x) + p \cdot e, -A] \bmod q)$;

$sk := Vector([1, s])$

$$pk := \begin{bmatrix} 21\,x^2 + 37\,x + 51 \\ 12\,x^2 + 100\,x + 88 \end{bmatrix}$$

$$sk := \begin{bmatrix} 1 \\ 78\,x^2 + 85\,x + 101 \end{bmatrix}$$

Encrypting messages, m1 and m2

$m1 := 4\,x^2 + 5\,x + 1$:

$m2 := 5\,x^2 + 2\,x + 4$:

$r1 := 62\,x^2 + 49\,x + 58$:

$r2 := 67\,x^2 + 49\,x + 14$:

$M := m1 + m2 \bmod p$;

$c1 := Vector([rem(pk[1] \cdot r1, Z, x) + m1, rem(pk[2] \cdot r1, Z, x)] \bmod q)$;

$c2 := Vector([rem(pk[1] \cdot r2, Z, x) + m2, rem(pk[2] \cdot r2, Z, x)] \bmod q)$;

$$M := 2\,x^2 + 5$$

$$c1 := \begin{bmatrix} 2\,x^2 + 93\,x + 56 \\ 27\,x^2 + 23\,x + 101 \end{bmatrix}$$

$$c2 := \begin{bmatrix} 69\,x^2 + 37\,x + 45 \\ 44\,x^2 + 78\,x + 34 \end{bmatrix}$$

Calculate C=c1+c2 mod q

$C := Vector(2)$:

$C := c1 + c2 \bmod q$

$$C := \begin{bmatrix} 71\,x^2 + 25\,x + 101 \\ 71\,x^2 + 101\,x + 30 \end{bmatrix}$$

Decrypting ciphertext C

$step1 := rem(expand(Multiply(Transpose(C), sk)), Z, x) \bmod q$;

$step2 := step1 \bmod p$

$$step1 := 2\,x^2 + 98\,x + 33$$

$$step2 := 2\,x^2 + 5$$

Figure I.3: Screenshot of Maple code of RLWE-CSCM homomorphic additions of two ciphertexts in Example I.3

Decryption is executed through two steps in (7.8).

Step 1:

$$[\langle C, sk\rangle]_q = C_1 + C_2 \cdot s \; mod \; q = 40\; x \; + \; 75. \tag{I.24}$$

Step2: modulo p operation is applied to (I.24),

$$m' = \left[[\langle C, sk\rangle]_q\right]_p = 40\; x \; + \; 75 \; mod \; 7 = 5x \; + 5, \tag{I.25}$$

We see that result (I.25) of decryption of (I.23) matches (I.22). Figure I.4 shows a screenshot of Maple code of Example I.4

Example I.5: Example of a single homomorphic multiplication

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys, A, s, and e are defined in (I.1)-(I.3). From (I.1)-(I.3), and (7.3), the public key $pk \in R_q^2$, and secret key, $sk \in R_q^2$ are defined in (I.4) and (I.5). respectively.

Encryption of the messages $m^{(1)}$ and $m^{(2)}$ in (I.15) and (I.16), using the random, $r^{(1)}$ and $r^{(2)}$ in (I.17) and (I.18) respectively, yielding $c^{(1)}$ and $c^{(2)}$ in (I.19) and (I.20). Let us calculate

$$m^{(1)} \cdot m^{(2)} = 3x^2 + 2x + 6 \in R_p\,, \tag{I.26}$$

Parameter Setup

$p := 7:$
$q := 15 \cdot p:$
$d := 3:$
$Z := x^d + 1:$

Gernerating Public key and Secret key

$A := 93x^2 + 110x + 17:$
$s := 78x^2 + 85x + 101:$
$e := 91x^2 + 58x + 62:$

$pk := Vector([rem(A \cdot s, Z, x) + p \cdot e, -A] \bmod q);$
$sk := Vector([1, s])$

$$pk := \begin{bmatrix} 21x^2 + 37x + 51 \\ 12x^2 + 100x + 88 \end{bmatrix}$$

$$sk := \begin{bmatrix} 1 \\ 78x^2 + 85x + 101 \end{bmatrix}$$

Encrypting messages, m1 and m2

$m1 := 4x^2 + 5x + 1:$
$m2 := 5x^2 + 2x + 4:$
$r1 := 62x^2 + 49x + 58:$
$r2 := 67x^2 + 49x + 14:$
$M := 1000 \cdot m1 + 2000 \cdot m2 \bmod p;$
$c1 := Vector([rem(pk[1] \cdot r1, Z, x) + m1, rem(pk[2] \cdot r1, Z, x)] \bmod q);$
$c2 := Vector([rem(pk[1] \cdot r2, Z, x) + m2, rem(pk[2] \cdot r2, Z, x)] \bmod q);$

$$M := 5x + 5$$

$$c1 := \begin{bmatrix} 2x^2 + 93x + 56 \\ 27x^2 + 23x + 101 \end{bmatrix}$$

$$c2 := \begin{bmatrix} 69x^2 + 37x + 45 \\ 44x^2 + 78x + 34 \end{bmatrix}$$

Calculate C=1000*c1+2000*c2 mod q

$C := Vector(2):$
$C := 1000 \cdot c1 + 2000 \cdot c2 \bmod q$

$$C := \begin{bmatrix} 35x^2 + 50x + 50 \\ 25x^2 + 80x + 55 \end{bmatrix}$$

Decrypting ciphertext C

$step1 := rem(expand(Multiply(Transpose(C), sk)), Z, x) \bmod q;$
$step2 := step1 \bmod p$

$$step1 := 40x + 75$$

$$step2 := 5x + 5$$

Figure I.4: Screenshot of Maple code of Example I.4

Let $M(c1, c2)$ be computed according to (7.23).

$$M(c1,c2)=\begin{pmatrix}45x^2\ +\ 29x\ +\ 19\\ 26x^2\ +\ 61x\ +\ 35\\ 16x^2\ +\ 17x\ +\ 1\end{pmatrix}\in R_q^2\,, \tag{I.27}$$

and the secret key, $sk^{M(c1,c2)}$, be found according to (7.25),

$$sk3=\begin{pmatrix}1\\ s\\ s^2\end{pmatrix}=\begin{pmatrix}1\\ 78x^2+85x+101\\ 91x^2+61x+91\end{pmatrix}. \tag{I.28}$$

Decryption is executed as follows

Step 1: Calculate $\langle M(c1,c2),sk3\rangle$ by (7.26). According to (7.8) and (7.23),

$$\begin{aligned}[\langle M(c1,c2),sk3\rangle]_q &= c_1^{(1)}c_1^{(2)}+\left(c_1^{(1)}c_2^{(2)}+c_2^{(1)}c_1^{(2)}\right)s+c_2^{(1)}c_2^{(2)}s^2\\ &=45x^2+44x+34.\end{aligned} \tag{I.29}$$

Step 2. By (I.29), (2.27),

$$\begin{aligned}\left[[\langle M(c1,c2),sk3\rangle]_q\right]_p &= 45x^2+44x+34\ mod\ p=3x^2\ +\ 2x\ +\ 6\\ &=m^{(1)}\cdot m^{(2)}.\end{aligned} \tag{I.30}$$

We see that result (I.30) of decryption of (I.27) matches (I.26). Figure 5 shows a screenshot of Maple code of Example 5.

Procedure of Homomorphic Multiplication

```
Mul := proc(a, b)
local C;
 C := Vector(3) :
 C[1] := rem(a[1]·b[1], Z, x) mod q :
 C[2] := rem(a[1]·b[2] + a[2]·b[1], Z, x) mod q :
 C[3] := rem(a[2]·b[2], Z, x) mod q :
return(C);
end proc:
```

Parameter Setup

$p := 7$:
$q := 15 \cdot p$:
$d := 3$:
$Z := x^d + 1$:

Gernerating Public key and Secret key

$A := 93x^2 + 110x + 17$:
$s := 78x^2 + 85x + 101$:
$e := 91x^2 + 58x + 62$:

$pk := Vector([rem(A \cdot s, Z, x) + p \cdot e, -A] \bmod q)$:
$sk := Vector([1, s])$:
$sk\_M\_c1\_c2 := Vector([1, s, rem(s^2, Z, x) \bmod q])$

$$sk\_M\_c1\_c2 := \begin{bmatrix} 1 \\ 78x^2 + 85x + 101 \\ 91x^2 + 61x + 91 \end{bmatrix}$$

Encrypting messages, m1 and m2

$m1 := 4x^2 + 5x + 1$:
$m2 := 5x^2 + 2x + 4$:
$r1 := 62x^2 + 49x + 58$:
$r2 := 67x^2 + 49x + 14$:

$M := rem(m1 \cdot m2, Z, x) \bmod p;$
$c1 := Vector([rem(pk[1] \cdot r1, Z, x) + m1, rem(pk[2] \cdot r1, Z, x)] \bmod q)$:
$c2 := Vector([rem(pk[1] \cdot r2, Z, x) + m2, rem(pk[2] \cdot r2, Z, x)] \bmod q)$:
$M\_c1\_c2 := Mul(c1, c2);$

$$M := 3x^2 + 2x + 6$$

$$M\_c1\_c2 := \begin{bmatrix} 45x^2 + 29x + 19 \\ 26x^2 + 61x + 35 \\ 16x^2 + 17x + 1 \end{bmatrix}$$

Decrypting ciphertext M_c1_c2 = c1* c2

$step1 := rem(expand(Multiply(Transpose(M\_c1\_c2), sk\_M\_c1\_c2)), Z, x) \bmod q;$
$step2 := step1 \bmod p$

$$step1 := 45x^2 + 44x + 34$$

$$step2 := 3x^2 + 2x + 6$$

Figure I.5: Screenshot of Maple code of Example I.5

Example I.6: Example of recrypting the product of ciphertexts encrypting $m^{(1)}$ and $m^{(2)}$ in (I.15) and (I.16).

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 3$, i.e., $R = \mathbb{Z}[x]/(x^3 + 1)$. Public and secret keys, A and e are defined in (I.1) and (I.3). Let β, γ, and s defined as follows,

$$\beta = 10, \gamma = 3, s = 30x^2 + 30x + 90, \tag{I.31}$$

From (I.1), (I.3), (I.31), and (7.3) the public key $pk \in R_q^2$, is defined as follows,

$$pk = \begin{pmatrix} 7x^2 + 46x + 74 \\ 12x^2 + 100x + 88 \end{pmatrix}, \tag{I.32}$$

Secret key sk, is defined according (7.40),

$$sk = \begin{pmatrix} 10 \\ 90x^2 + 90x + 60 \end{pmatrix} \in R_q. \tag{I.33}$$

In order to encrypt messages encrypting $m^{(1)}$ and $m^{(2)}$ in (I.15) and (I.16), $K = 5$ is used to pad the messages according to (7.37)

$$mK^{(1)} \leftarrow K \cdot m^{(1)} = 6x^2 + 4x + 5 \in R_p, \tag{I.34}$$

$$mK^{(2)} \leftarrow K \cdot m^{(2)} = 4x^2 + 3x + 6 \in R_p. \tag{I.35}$$

Let's find the product of $M^{(1)}$, $M^{(2)}$,

$$mK^{(1)} \cdot mK^{(2)} = 5x^2 + x + 3 \in R_p. \tag{I.36}$$

Encryption of $M^{(1)}, M^{(2)}$ is performed according to (7.5),

$$c^{(1)} \leftarrow Enc_{pk,r^{(1)}}\left(mK^{(1)}\right) = \begin{pmatrix} 9x^2 + 89x + 52 \\ 27\,x^2 + 23\,x + 101 \end{pmatrix}, \tag{I.37}$$

$$c^{(2)} \leftarrow Enc_{pk,r^{(2)}}\left(mK^{(2)}\right) = \begin{pmatrix} 69x^2 + 24x + 32 \\ 44\,x^2 + 78\,x + 34 \end{pmatrix}. \tag{I.38}$$

where $r^{(1)}$ and $r^{(2)}$in (I.17) and (I.18). The product $M\left(c^{(1)}, c^{(2)}\right)$ is computed according to (7.23),

$$M\left(c^{(1)}, c^{(2)}\right) = \begin{pmatrix} 27\,x^2 + 10\,x + 32 \\ 71\,x^2 + 3\,x + 37 \\ 16\,x^2 + 17\,x + 1 \end{pmatrix}. \tag{I.39}$$

Recryption key, $k_{rec}$is defined according to (7.42),

$$k_{rec} = \begin{pmatrix} 80 \\ 90\,x^2 + 90\,x + 60 \\ 60\,x \end{pmatrix} \in R_q^3. \tag{I.40}$$

Recryption process of $M\left(c^{(1)}, c^{(2)}\right)$ is performed by Algorithm 7.1,

1- $Mc1_2 \leftarrow M\left(c^{(1)}, c^{(2)}\right)$

2- $mK' \leftarrow Dec_{k_{rec}}(Mc1_2) = \left[[\langle M(c1, c2), sk3\rangle]_q (K\beta^2)^{-1}\right]_p$

=$[(15x^2 + 80x + 100)\,(5)]_p = 5x^2 + x + 3 = M^{(1)}M^{(2)}$.

3- $C\_K\_m1\_2 \leftarrow Enc_{pk,r^3}(mK') = \begin{pmatrix} 90\,x^2 + 8\,x + 30 \\ 43\,x^2 + 11\,x + 73 \end{pmatrix}$, where $r^{(3)} =$

$103x^2 + 39x + 17$.

Retrieving the product of $m^{(1)}$ and $m^{(2)}$is done by executing Algorithm 7.2,

1- $tmp \leftarrow Dec_{sk3}(C\_K\_m1\_2) = \left[\left[\langle C_{K_{m1_2}}, sk\rangle\right]_q \gamma^{-1}\right]_p$
=$[(15x^2 + 80x + 30)(5)]_p = 5x^2 + x + 3$
2- $m' \leftarrow tmp \cdot (K^{-1})^2 \bmod p = (5x^2 + x + 3)(3)^2 \bmod 7$
$= 3x^2 + 2x + 6 = m^{(1)} \cdot m^{(2)}$ in (I.26)

Figure I.6 shows a screenshot of Maple code of Example I.6.

Example I.7: Example of calculating 512 multiplications homomorphically of $m^{(1)}$ in Example I.6 using Power2Exponent.

Let the message m from (I.15) be encrypted by conditions of Example I.6. Thus, the encryption of the padded message is (I.37).

Let's calculate m^{pwr}, $pwr = 512$.

$$m^{pwr} = 5\,x^2 + x + 3 \in R_p. \tag{I.41}$$

In order to compute m^{pwr}, first, Algorithm 3 is executed and return $C = mK^{pwr}$. Our Maple implementation computed the result (I.39) in less than 5 milliseconds.

$$C = mK^{pwr} = \begin{pmatrix} 68\,x^2 + 78\,x + 87 \\ 39\,x^2 + 89\,x + 50 \end{pmatrix}. \tag{I.42}$$

Extracting m^{pwr} out of C (I.42) is done by executing Algorithm 4,

1- $tmp \leftarrow Dec_{sk}(C) = 6x^2 + 4x + 5$
2- $m' \leftarrow tmp \cdot (K^{-1})^{pwr} \bmod p = 5x^2 + x + 3 = m^{pwr}$

Figure 9 shows Maple code of Example 7.

Padding messages, m1 and m2and Encrypting

$m1 := 4x^2 + 5x + 1$:

$m2 := 5x^2 + 2x + 4$:

$K := 5$:

$M1 := K \cdot m1 \bmod p$;

$M2 := K \cdot m2 \bmod p$;

$$6x^2 + 4x + 5$$

$$4x^2 + 3x + 6$$

Computing the product M1*M2

$M := rem(M1 \cdot M2, Z, x) \bmod p$;

$$5x^2 + x + 3$$

Encrypting padded messages M1 and M2

$r1 := 62x^2 + 49x + 58$:

$r2 := 67x^2 + 49x + 14$:

$c1 := Vector([rem(pk[1] \cdot r1, Z, x) + M1, rem(pk[2] \cdot r1, Z, x)] \bmod q)$;

$c2 := Vector([rem(pk[1] \cdot r2, Z, x) + M2, rem(pk[2] \cdot r2, Z, x)] \bmod q)$;

$$\begin{bmatrix} 9x^2 + 89x + 52 \\ 27x^2 + 23x + 101 \end{bmatrix}$$

$$\begin{bmatrix} 69x^2 + 24x + 32 \\ 44x^2 + 78x + 34 \end{bmatrix}$$

Computing M(c1,c2)

$M\_c1\_c2 := Mul(c1, c2)$;

$$\begin{bmatrix} 27x^2 + 10x + 32 \\ 71x^2 + 3x + 37 \\ 16x^2 + 17x + 1 \end{bmatrix}$$

Decrypting ciphertext M_c1_c2 = M1* M2 using krec

$krec := K \cdot sk3 \bmod q$;

$igcdex(krec[1], p, 'krec1Inv', 'a')$: $krec1Inv := krec1Inv \bmod p$:

$step1 := rem(expand(Multiply(Transpose(M\_c1\_c2), krec)), Z, x) \bmod q$;

$M := step1 \cdot krec1Inv \bmod p$;

$$\begin{bmatrix} 80 \\ 90x^2 + 90x + 60 \\ 60x \end{bmatrix}$$

$$15x^2 + 80x + 100$$

$$5x^2 + x + 3$$

Encrypting the message, M

$r3 := 103x^2 + 39x + 17$:

$C\_k\_m1\_2 := Vector([rem(pk[1] \cdot r3, Z, x) + M, rem(pk[2] \cdot r3, Z, x)] \bmod q)$;

$$\begin{bmatrix} 90x^2 + 8x + 30 \\ 43x^2 + 11x + 73 \end{bmatrix}$$

Decrypting the ciphertext C_k_m1_2

$step1 := (rem(expand(Multiply(Transpose(C\_k\_m1\_2), sk)), Z, x) \bmod q)$;

$tmp := (step1 \cdot gama^{-1}) \bmod p$;

$$15x^2 + 80x + 30$$

$$5x^2 + x + 3$$

Retrieving m1*m2 out of tmp

$igcdex(K, p, 'kinv', 'a')$: $kinv$:

$m := tmp \cdot kinv^2 \bmod p$

$$3x^2 + 2x + 6$$

Computing m1*m2

$m1m2 := (rem(expand(m1 \cdot m2), Z, x) \bmod q) \bmod p$

$$3x^2 + 2x + 6$$

Figure I.6: Screenshot of Maple code of Example I.6

Padding and Encrypting the message, m

$m := 4x^2 + 5x + 1$:

$K := 5$:

$M := K \cdot m \bmod p$;

$$6x^2 + 4x + 5$$

Computing m^512

$expand(rem(m^{512}, Z, x)) \bmod p$

$$5x^2 + x + 3$$

Encrypting padded messages M

$r := 62x^2 + 49x + 58$:

$c := Vector([rem(pk[1] \cdot r, Z, x) + M, rem(pk[2] \cdot r, Z, x)]) \bmod q$;

$$\begin{bmatrix} 9x^2 + 89x + 52 \\ 27x^2 + 23x + 101 \end{bmatrix}$$

Generating k_rec

$pwr := 512$:

$k\_rec := K \cdot sk3 \bmod q$;

$$\begin{bmatrix} 80 \\ 90x^2 + 90x + 60 \\ 60x \end{bmatrix}$$

Computing c^512

$r\_rec := 45x^2 + 93x + 62$:

$st := time()$:

for *iter* **from** 1 **to** 1000 **do** $C := Power2Exponent(c, pwr, k\_rec, pk, r\_rec)$ **end do**:

$time() - st$;

C;

$$4.579$$

$$\begin{bmatrix} 68x^2 + 78x + 87 \\ 39x^2 + 89x + 50 \end{bmatrix}$$

Finding the multiplicative invrse of K

$igcdex(K, p, 'Kinv', 'a') : Kinv := Kinv \bmod p$;

$$3$$

Decrypting ciphertext C:=Enc(m^512) using *DecPower2Exponent*

$DecPower2Exponent(C, sk, Kinv, pwr)$

$$5x^2 + x + 3$$

Figure I.7: Screenshot of Maple code of Example I.7

Example I.8: Example of Successful Modulo c_2 Attack.

Let $p = 7$, $q = 15 \cdot p = 105$, and $d = 7$, i.e., $R = \mathbb{Z}[x]/(x^7 + 1)$. Public and secret keys, $A, s,$ and e are defined in (I.1)-(I.3).

From (I.1)-(I.3) and (7.3), the public key $pk \in R_q^2$, is defined as follows:

$$pk = \begin{pmatrix} A \cdot s + p \cdot e \\ -A \end{pmatrix} = \begin{pmatrix} 9x^4 + 21x^2 + 46x + 51 \\ 12x^2 + 100x + 88 \end{pmatrix}. \quad \text{(I.43)}$$

Encryption of the message,

$$m = 5x^3 + x^2 + 1 \in R_p, \quad \text{(I.44)}$$

using random,

$$r = 67x^2 + 49x + 14 \in R_q, \quad \text{(I.45)}$$

is performed according to (7.5),

$$c = \begin{pmatrix} 78x^6 + 21x^5 + 63x^4 + 21x^3 + 86x^2 + 98 + 85 \\ 69x^4 + 43x^3 + 44x^2 + 42x + 77 \end{pmatrix} \in R_q^2. \quad \text{(I.46)}$$

From (I.46), $\deg(pk_1 \cdot r + m) = 6 < 7 = d$ meeting (7.57), and $\deg(pk_2 \cdot r) = 4 > 3 = \deg(m)$ meeting (7.58). Thus, message can be revealed by the COA attack steps:

Step 1: Applying $mod\ c_2$ operation to c_1, from (I.46)

$$c_1 mod\ c_2 = -\frac{3050639}{109503}x^3 - \frac{401434}{109503}x^2 + \frac{4024783}{36501}x + \frac{5584805}{109503} \quad \text{(I.47)}$$

Step 2: Applying $mod\ p$ operation to (I.47),

$$m' = (c_1 mod\ c_2) mod\ p\ = 3x^3 + x^2 + 1 = m \quad \text{(I.48)}$$

Figure I.10 shows a screenshot of Maple code of Example I.8

```
Procedure of Power2Exponent
Power2Exponent := proc(c, pwr, k_rec, pk, r_rec)
local C, Dec_tmp, tmp, i;
C := c;
i := pwr;
while(i > 1) do
  tmp := Mul(C, C) :
  Dec_tmp := ((rem(expand(Multiply(Transpose(tmp), k_rec)), Z, x) mod q) · k_rec[1]^-1) mod p;
 C := Vector([rem(pk[1]·r_rec, Z, x) + Dec_tmp, rem(pk[2]·r_rec, Z, x) ] mod q);
 i := i/2;
end do:
return( C);
end proc:
```

Figure I.8: Maple code implementation of Power2Exponent

```
Procedurce of DecPower2Exponent
DecPower2Exponent := proc( C, sk, Kinv, pwr )
 local temp, m";
temp := (rem( expand( Multiply( Transpose( C ), sk) ) , Z, x) mod q ) mod p;
 m" := ( Kinv^pwr · temp ) mod p :
 return m";
 end proc:
```

Figure I.9: Maple code implementation of DecPower2Exponent

Parameter Setup

$p := 7$:

$q := 15 \cdot p$:

$d := 7$:

$Z := x^d + 1$:

Gernerating Public key and Secret key

$Ap := 93x^2 + 110x + 17$:

$sp := 78x^2 + 85x + 101$:

$e := 91x^2 + 58x + 62$:

$pk := Vector([rem(Ap \cdot sp, Z, x) + p \cdot e, -Ap] \bmod q)$;

$s := 78x^2 + 85x + 101$:

$sk := Vector([1, s])$

$$\begin{bmatrix} 9x^4 + 21x^2 + 46x + 51 \\ 12x^2 + 100x + 88 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 78x^2 + 85x + 101 \end{bmatrix}$$

Encrypting message, m

$m := 5x^3 + x^2 + 1$:

$r := 67x^2 + 49x + 14$:

$c := Vector([rem(pk[1] * r, Z, x) + m, rem(pk[2] * r, Z, x)]) \bmod q$;

$$\begin{bmatrix} 78x^6 + 21x^5 + 63x^4 + 21x^3 + 86x^2 + 98x + 85 \\ 69x^4 + 43x^3 + 44x^2 + 42x + 77 \end{bmatrix}$$

Attacking ciphertext c

$step1 := rem(c[1], c[2], x)$;

$step2 := step1 \bmod p$;

m;

$$-\frac{3050639}{109503}x^3 - \frac{401434}{109503}x^2 + \frac{4024783}{36501}x + \frac{5584805}{109503}$$

$$5x^3 + x^2 + 1$$

$$5x^3 + x^2 + 1$$

Encrypting message, m

$m := 5x^3 + x^2 + 1$:

$r := 67x^2 + 49x + 14$:

$c := Vector([rem(pk[1] * r, Z, x) + m, rem(pk[2] * r, Z, x)]) \bmod q$;

$$\begin{bmatrix} 78x^6 + 21x^5 + 63x^4 + 21x^3 + 86x^2 + 98x + 85 \\ 69x^4 + 43x^3 + 44x^2 + 42x + 77 \end{bmatrix}$$

Attacking ciphertext c

$step1 := rem(c[1], c[2], x)$;

$step2 := step1 \bmod p$;

m;

$$-\frac{3050639}{109503}x^3 - \frac{401434}{109503}x^2 + \frac{4024783}{36501}x + \frac{5584805}{109503}$$

$$5x^3 + x^2 + 1$$

$$5x^3 + x^2 + 1$$

Figure I.10: A screenshot of Maple code of Example I.8.

Example I.9: Example of Failing Modulo c_2 Attack.

Let p, q, and d from Example I.8. And let, A, s, and e are defined in (I.1)-(I.3). Public key $pk \in R_q^2$, is defined in (I.43).

Encryption of the message, m, in (I.44) is performed using random, r with degree= 1, satisfying (I.6),

$$r = 49x + 14 \in R_q. \tag{I.49}$$

Encryption is performed according to (7.5),

$$c = \begin{pmatrix} 73x^8 + 10x^6 + 47x^5 + 81x^4 + 30x^3 + 60x^2 + 35x + 102 \\ 11x^5 + 3x + 5 \end{pmatrix} \in R_q^2. \tag{I.50}$$

From (7.5) and (7.6), $\deg(c_2) = \deg(pk_2 \cdot r) = 3 \leq 3 = \deg(m)$ meeting (7.6). Thus, message can't be revealed by the COA attack steps:

Step 1: Applying $mod\ c_2$ operation to c_1, from (I.50)

$$c_1 mod\ c_2 = -\frac{8620}{243}x^2 + \frac{3013}{81}x - \frac{80}{243} \tag{I.51}$$

Step2: Applying $mod\ p$ operation to (I.51)

$$m' = (c_1 mod\ c_2) mod\ p\ = 5x^2 + 6x + 5 \neq m \tag{I.52}$$

Figure I.11 shows a screenshot of Maple code of Example I.9.

Parameter Setup

$p := 7:$
$q := 15 \cdot p:$
$d := 7:$
$Z := x^d + 1:$

Gernerating Public key and Secret key

$Ap := 93x^2 + 110x + 17:$
$sp := 78x^2 + 85x + 101:$
$e := 91x^2 + 58x + 62:$
$pk := Vector([rem(Ap \cdot sp, Z, x) + p \cdot e, -Ap] \bmod q);$

$s := 78x^2 + 85x + 101:$
$sk := Vector([1, s])$

$$\begin{bmatrix} 9x^4 + 21x^2 + 46x + 51 \\ 12x^2 + 100x + 88 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 78x^2 + 85x + 101 \end{bmatrix}$$

Encrypting message, m

$m := 5x^3 + x^2 + 1:$
$r := 49x + 14:$

$c := Vector([rem(pk[1] * r, Z, x) + m, rem(pk[2] * r, Z, x)]) \bmod q;$

$$\begin{bmatrix} 21x^5 + 21x^4 + 89x^3 + 29x^2 + 98x + 85 \\ 63x^3 + 28x^2 + 42x + 77 \end{bmatrix}$$

Attacking ciphertext c

$step1 := rem(c[1], c[2], x);$
$step2 := step1 \bmod p;$
$m;$

$$-\frac{8620}{243}x^2 + \frac{3013}{81}x - \frac{80}{243}$$

$$5x^2 + 6x + 5$$

$$5x^3 + x^2 + 1$$

Figure I.11: A screenshot of Maple code of Example I.8.

Appendix J: Source Code of RCPKC Performance Tests

```
#define
_CRT_SECUR
E_NO_DEPRE
CATE
            #include <stdio.h>
            #include <NTL/ZZ.h>
            #include <fstream>
            #include <iostream>
            #include <iomanip>
            #include <Windows.h>

            using namespace std;
            using namespace NTL;



            int main()
            {
                                    double        pcFreq;        // Counter
            frequency (timer resolution)
                                    __int64       counterStart;  // Timer
            value
                                    LARGE_INTEGER li;            // Large
            interger for timer value
                                    char
                                      str[255];      // String for name
                                    double        elapsed;       // Elapsed
            time in seconds
                                    int           retcode;       // Return
            code
                                    retcode = QueryPerformanceFrequency(&li);
                                    if (retcode == 0)
                                                                   printf("***
            ERROR - QueryPerformanceFrequency() failed \n");
                                    pcFreq = li.QuadPart / 1000000000.0;
                                    double st, ft;
                                    /* Fix qLen = 473, mgLen = 225 */
                                    long mgLen = 225, qLen = 473;
                                    ZZ two = conv<ZZ>("2"), f, g, m, r, e, a,
            q, h, Fg, ans;
                                    // Fix RCPKC parameters
                                    q = power(two, qLen);
                                    g = power(two, mgLen) - 1;
                                    f = power(two, qLen - mgLen - 1) - 1;
```

```
                        r =
conv<ZZ>("741291310836658073589740569582185351660402929503737800217285839760148934");
                        h =
conv<ZZ>("121943302746718446538343641788795558818304614947850435580435818735368345111350811934548310975825265757390606062845204110669020756287780993105 93");
                        Fg =
conv<ZZ>("365263789408006274939445144047909989680128334658003361567844049456 21");
                        FILE *fptr = fopen("ResultsFile.csv",
"w");
                        ifstream
in("Message_samples3.csv");//Select Message_samples3.csv,
Message_samples4.csv, Message_samples5.csv for 10^3, 10^4, 10^5
plaintext messages respectively

                        while (!in.eof()) {
                                                in.getline(str,
255);
                                                m =
conv<ZZ>(str);

                        QueryPerformanceCounter(&li);
                                                counterStart =
li.QuadPart;
                                                rem(e, h*r + m,
q); //RCPKC encryption

                        QueryPerformanceCounter(&li);
                                                elapsed =
((li.QuadPart - counterStart) / pcFreq);
                                                fprintf(fptr,
"%f%s", elapsed, ",");

                        QueryPerformanceCounter(&li);
                                                counterStart =
li.QuadPart;
                                                rem(a, f*e, q);
//Step 1 of RCPKC Decryption
                                                rem(ans, Fg*a,
g); //Step 2 of RCPKC Decryption

                        QueryPerformanceCounter(&li);
                                                elapsed =
((li.QuadPart - counterStart) / pcFreq);
```

```
                                                    fprintf(fptr,
"%f\n", elapsed);
                        }
                        return 0;
}
```