2019-08-29

Improved Efficiency of a Linearly Homomorphic Cryptosystem

Das, Parthasarathi

UNIVERSITY OF CALGARY

Improved Efficiency of a Linearly Homomorphic Cryptosystem

by

Parthasarathi Das

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

AUGUST, 2019

Abstract

We present an extended version of the Castagnos and Laguillaumie linearly homomorphic cryptosystem in which the non-maximal imaginary quadratic order is allowed to have conductor equal to a power of a product of distinct primes as opposed to a single prime. Numerical results obtained with an optimized C implementation demonstrate that this variation improves performance when large messages and exponents are used. When compared to the cryptosystems of Paillier and Bresson *et al.* at the same security levels, the basic version of Castagnos and Laguillaumie is the fastest at high security levels for small messages.

Acknowledgements

I would like to thank my supervisor, Dr. Michael John Jacobson, Jr., who suggested this project to me. Without his kindness, this thesis would never have seen the light of day. I would also like to thank my co-supervisor, Dr. Renate Scheidler, for her continued support and guidance during this learning process. I would like to thank them both for all their valuable advice and lessons and for making my time as a Master's student a very fulfilling and rewarding experience.

In addition, I would like to thank my examining committee, Dr. Robin Cockett and Dr. Dang Khoa Nguyen, for taking the time to read this thesis and for their valuable comments.

I would like to thank my colleagues, Randy Yee, Sebastian Lindner, Ha Thanh Nguyen Tran, Randall Apperley and Christian Bagshaw for all their help and valuable conversations and time spent together during my time at the University of Calgary.

Finally, I would like to thank my family for their support and all my friends and colleagues for their presence in my life. Thank you all.

To mom and dad...

Table of Contents

Abstract ii

Acknowledgements iii

Dedication iv

Table of Contents v

List of Figures vii

List of Algorithms ix

List of Tables x

List of Symbols, Abbreviations and Nomenclature xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 4
- 1.3 Organisation of Thesis . . . 5

2 Background 6
- 2.1 Imaginary Quadratic Fields . . . 6
  - 2.1.1 Orders in $\mathbb{K}$ . . . 8
  - 2.1.2 Ideals in $\mathcal{O}$ . . . 10
  - 2.1.3 Ideals Prime to f . . . 13
- 2.2 Ideal Class Group . . . 14
  - 2.2.1 A Particular Subgroup of $C(\mathcal{O}_{\Delta_f})$ . . . 16
  - 2.2.2 Cohen-Lenstra Heuristics . . . 17
  - 2.2.3 Discrete Logarithm in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ . . . 18
  - 2.2.4 The 2-Sylow Subgroup . . . 19

3 Castagnos and Laguillaumie Encryption Scheme 21
- 3.1 Introduction . . . 21
- 3.2 Generic Castagnos and Laguillaumie Construction . . . 23
- 3.3 Instantiation in Class Groups . . . 27

3.3.1 Security . . . 30
3.4 Short Cipher Variant . . . 33
3.5 Handling Longer Message Lengths . . . 35
3.6 Conductor Choices . . . 38
3.7 Summary . . . 38

4 Using Conductor $f = (p_1 p_2 \cdots p_N)^t$ 40
4.1 Introduction . . . 40
4.2 $N \geq 1$ and $t = 1$ . . . 40
4.3 $N \geq 1$ and $t > 1$ . . . 44
4.4 Parameter Choices . . . 47
4.4.1 Restrictions on Prime Factors of f . . . 48
4.4.2 Selection of Random Exponents . . . 50
4.5 Summary . . . 50

5 Results 52
5.1 Description of Experiments . . . 53
5.1.1 Objectives . . . 53
5.1.2 Methodology . . . 53
5.2 Analysis of CL Schemes . . . 56
5.2.1 Case $bl(f) < bl(f)_{\max}$ . . . 57
5.2.2 Case $bl(f) > bl(f)_{\max}$ . . . 67
5.2.3 Summary . . . 68
5.3 Comparison of CL, Paillier and BCP schemes . . . 73

6 Conclusion 82
6.1 Future Work . . . 83

Bibliography 85

A BCP Runtimes 92

B Paillier Runtimes 93

C CL Runtimes - LongCipher 94

D CL Runtimes - ShortCipher 111

List of Figures

5.1 LongCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 58
5.2 ShortCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 59
5.3 LongCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 59
5.4 ShortCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 60
5.5 LongCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 60
5.6 ShortCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 61
5.7 LongCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 61
5.8 ShortCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 62
5.9 LongCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 63
5.10 ShortCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 64
5.11 LongCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 64
5.12 ShortCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 65
5.13 LongCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 65
5.14 ShortCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents 66
5.15 LongCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 66
5.16 ShortCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents 67
5.17 LongCipher Encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 68

5.18 ShortCipher Encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 69
5.19 LongCipher Decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 69
5.20 ShortCipher Decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 70
5.21 LongCipher and ShortCipher Encryption-CRT 3 runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 70
5.22 LongCipher and ShortCipher Decryption-CRT 3 runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents 71
5.23 Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 128-bit security level 76
5.24 Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 128-bit security level 77
5.25 Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 192-bit security level 78
5.26 Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 192-bit security level 79
5.27 Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 256-bit security level 80
5.28 Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 256-bit security level 81

List of Algorithms

2.1.1 FindIdealPrimeTo – [24, Algorithm 8] . . . 15
2.1.2 GoToNonMaxOrder – [24, Algorithm 9] . . . 15
2.1.3 GoToMaxOrder – [24, Algorithm 10] . . . 16
3.2.1 KeyGen . . . 24
3.2.2 Encrypt . . . 25
3.2.3 Decrypt . . . 25
3.2.4 EvalSum . . . 26
3.2.5 EvalScal . . . 27
3.3.1 Gen . . . 29
3.3.2 Solve . . . 29
3.4.1 Gen – ShortCipher . . . 34
3.4.2 Encrypt – ShortCipher . . . 34
3.4.3 Decrypt – ShortCipher . . . 35
3.5.1 Solve – Lift . . . 38
4.2.1 Gen – using $f = (p_1 p_2 \cdots p_N)^t$ . . . 41
4.2.2 Decrypt – CRT 1 . . . 42
4.2.3 Encrypt – CRT 2 . . . 43
4.2.4 Decrypt – CRT 2 . . . 43
4.2.5 Encrypt – CRT 3 . . . 44
4.2.6 Decrypt – CRT 3 . . . 44
4.3.1 Decrypt – using Pohlig-Hellman . . . 45
4.3.2 Encrypt – CRT 3 . . . 46
4.3.3 Decrypt – CRT 3 . . . 46

List of Tables

3.1 Parameter sizes (in bits) . . . 33
3.2 Number of ideal exponentiation steps in each variant . . . 39

4.1 Parameter sizes (in bits) . . . 47
4.2 Number of ideal exponentiations in each variant . . . 51

5.1 Summary of Parameter Choices . . . 55
5.2 Summary of Exponentiation Algorithms used in CL schemes . . . 56
5.3 Description of headers and labels in Table 5.4 and Table 5.5 . . . 71
5.4 Summary of the CL cryptosystem with short exponent lengths (in ms) . . . 72
5.5 Summary of the CL cryptosystem with full exponent lengths(in ms) . . . 72
5.6 Summary of best performance (in ms) — 128-bit security . . . 73
5.7 Summary of best performance (in ms) — 192-bit security . . . 74
5.8 Summary of best performance (in ms) — 256-bit security . . . 74

A.1 Performance of BCP scheme . . . 92

B.1 Performance of Paillier scheme . . . 93

C.1 LongCipher, 1828, 16, short length . . . 95
C.2 LongCipher, 1828, 16, full length . . . 95
C.3 LongCipher, 1828, 80, short length . . . 95
C.4 LongCipher, 1828, 80, full length . . . 96
C.5 LongCipher, 1828, 256, short length . . . 97
C.6 LongCipher, 1828, 256, full length . . . 98
C.7 LongCipher, 1828, 3072, short length . . . 99
C.8 LongCipher, 1828, 3072, full length . . . 100
C.9 LongCipher, 3598, 16, short length . . . 100
C.10 LongCipher, 3598, 16, full length . . . 100
C.11 LongCipher, 3598, 80, short length . . . 101
C.12 LongCipher, 3598, 80, full length . . . 101
C.13 LongCipher, 3598, 256, short length . . . 102
C.14 LongCipher, 3598, 256, full length . . . 103
C.15 LongCipher, 3598, 7680, short length . . . 104
C.16 LongCipher, 3598, 7680, full length . . . 105
C.17 LongCipher, 5972, 16, short length . . . 105
C.18 LongCipher, 5972, 16, full length . . . 105

C.19 LongCipher, 5972, 80, short length 106
C.20 LongCipher, 5972, 80, full length 106
C.21 LongCipher, 5972, 256, short length 107
C.22 LongCipher, 5972, 256, full length 108
C.23 LongCipher, 5972, 15360, short length 109
C.24 LongCipher, 5972, 15360, full length 110

D.1 ShortCipher, 1828, 16, short length 111
D.2 ShortCipher, 1828, 16, full length 111
D.3 ShortCipher, 1828, 80, short length 112
D.4 ShortCipher, 1828, 80, full length 112
D.5 ShortCipher, 1828, 256, short length 113
D.6 ShortCipher, 1828, 256, full length 114
D.7 ShortCipher, 1828, 3072, short length 115
D.8 ShortCipher, 1828, 3072, full length 116
D.9 ShortCipher, 3598, 16, short length 116
D.10 ShortCipher, 3598, 16, full length 116
D.11 ShortCipher, 3598, 80, short length 117
D.12 ShortCipher, 3598, 80, full length 117
D.13 ShortCipher, 3598, 256, short length 118
D.14 ShortCipher, 3598, 256, full length 119
D.15 ShortCipher, 3598, 7680, short length 120
D.16 ShortCipher, 3598, 7680, full length 121
D.17 ShortCipher, 5972, 16, short length 121
D.18 ShortCipher, 5972, 16, full length 121
D.19 ShortCipher, 5972, 80, short length 122
D.20 ShortCipher, 5972, 80, full length 122
D.21 ShortCipher, 5972, 256, short length 123
D.22 ShortCipher, 5972, 256, full length 124
D.23 ShortCipher, 5972, 15360, short length 125
D.24 ShortCipher, 5972, 15360, full length 126

List of Symbols, Abbreviations and Nomenclature

Symbol or abbreviation	Definition
$\mathbb{N}$	the natural numbers
$\mathbb{Z}$	the integers
$\mathbb{Q}$	the rational numbers
$\mathbb{R}$	the real numbers
$\mathbb{C}$	the complex numbers
$\mathbb{K}$	an algebraic number field
$\mathcal{O}$	an order of $\mathbb{K}$
$\mathcal{O}_{\Delta_{\mathbb{K}}}$	the maximal order of $\mathbb{K}$
Δ	the discriminant of the order $\mathcal{O}$
$\Delta_{\mathbb{K}}$	the fundamental discriminant of $\mathbb{K}$
f	the conductor of $\mathcal{O}$
$\mathcal{O}_{\Delta}$	the order of discriminant Δ
$\mathfrak{a}$	an ideal of an order $\mathcal{O}$
$[\mathfrak{a}]$	the set of all $\mathcal{O}$-ideals equivalent to $\mathfrak{a}$
$C(\mathcal{O}_{\Delta})$	the class group of $\mathcal{O}_{\Delta}$
$C(\mathcal{O}_{\Delta_{\mathbb{K}}})$	the class group of maximal order $\mathcal{O}_{\Delta_{\mathbb{K}}}$
$h(\Delta_{\mathbb{K}})$	the class number of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$
$(\mathbb{Z}/n\mathbb{Z})$	the set of integers $\{0, 1, \cdots, n-1\}$
$(\mathbb{Z}/n\mathbb{Z})^{\star}$	the set of units in $(\mathbb{Z}/n\mathbb{Z})$
DDH	Decisional Diffie-Hellman
DL	Discrete Logarithm
CRT	Chinese Remainder Theorem
PDL	Partial Discrete Logarithm
CL	Castagnos and Laguillaumie cryptosystem
BCP	Bresson *et al.* cryptosystem

Chapter 1

Introduction

1.1 Motivation

Homomorphic encryption is a type of encryption mechanism that allows computations to be performed on ciphertexts in a way such that the result of these computations, a ciphertext itself, is the encryption of the same computations performed on the corresponding plaintexts. The term *homomorphic* refers to the concept of a homomorphism in algebra; the encryption and decryption functions act as homomorphisms between the plaintext and ciphertext spaces. The homomorphic operation can either be Boolean or arithmetic circuits. Armknecht *et al.* discussed the various types of homomorphic encryptions in [3]. Typically, we can categorise homomorphic encryption into partially homomorphic, somewhat homomorphic, levelled fully homomorphic and fully homomorphic encryption schemes.

Partially homomorphic encryption consists of schemes which support the evaluation of circuits consisting of only one type of logic gate such as addition or multiplication. Somewhat homomorphic encryption consists of schemes which support the evaluation of two types of logic gates but only for a subset of all the possible circuits using the two gates. Leveled fully homomorphic encryption consists of schemes which support the evaluation of arbitrary circuits whose depth has been pre-determined *i.e.*, has an upper bound. Fully homomorphic

encryption is the strongest notion of homomorphic encryption in the sense that it supports the evaluation of arbitrary circuits of unbounded depth.

Since homomorphic encryption schemes allow meaningful computations to be performed on the ciphertexts, they are malleable by nature. In [5], Bellare *et al.* show that malleability precludes an encryption scheme from achieving the highest level of security, indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). However, some of the most widely deployed encryption schemes such as RSA [40], Elgamal [20] and Paillier [35] are some type of homomorphic in nature. More specifically, the Paillier cryptosystem is a partially homomorphic encryption scheme that supports addition over the plaintext and ciphertext spaces. In other words, it is *additively* homomorphic. By definition, an additively homomorphic encryption scheme also satisfies multiplication by a scalar *i.e.*, a ciphertext can be multiplied by a scalar and when this result is decrypted, we get the product of the corresponding plaintext and the scalar. These type of schemes are called *linearly homomorphic* under addition as they support linear combinations of additions in the plaintext and ciphertext spaces.

Thus, a linearly homomorphic cryptosystem is one for which linear combinations of ciphertexts can be computed in such a way that the result is the encryption of the same linear combination of the corresponding plaintexts. Such cryptosystems have a number of applications. For example, when used for electronic voting, encrypted votes (encrypting 1 for 'yes' and 0 for 'no') can be tallied with a single decryption by homomorphically adding the ciphertexts and decrypting the result. Two well-known recent examples of linearly homomorphic encryption systems are due to Paillier [35] and Bresson *et al.* [8]. In both cases, the security relies on the presumed intractability of integer factorisation whereas the latter also relies on the difficulty of the discrete logarithm (DL) problem.

In [12], Castagnos and Laguillaumie presented a linearly homomorphic encryption scheme whose security is based on the hardness of the decision Diffie-Hellman (DDH) problem in a group that has a subgroup in which the discrete logarithm problem can be solved easily; this setting is referred to as a 'DDH group with an easy DL subgroup'. Assuming the existence

of such groups, they described a linearly homomorphic encryption scheme and a function encryption scheme that is provably one-way and semantically secure subject to relatively standard hardness assumptions. They also gave an instantiation of their cryptosystem using the ideal class group of a non-maximal imaginary quadratic order with prime conductor as the DDH group with easy DL subgroup. Subsequently, this cryptosystem was used in combination with a variant of Elgamal in an encryption switching protocol [10], providing an efficient setting for a secure two-party computation protocol.

The cryptosystem of [12] has two main novel features. Firstly, it is the only purely linearly homomorphic cryptosystem (not counting the fully homomorphic cryptosystems based on the learning with errors problem) whose security does not depend on integer factorization — all hardness assumptions are versions of Diffie-Hellman and discrete logarithm problems. The second feature is that the size of the message space can be chosen independently of the security parameter. This is especially attractive in electronic voting applications, as the message space can be chosen just large enough to handle the required number of votes. The voters vote either a 'yes' or 'no' for a candidate. The votes get encrypted by encrypting 1 for 'yes' and 0 for 'no'. The election manager receives all the ciphertexts $c_1, c_2 \cdots c_n$ and computes their product c. They then decrypt c and receive the sum of all votes for a candidate. The advantage of this system is that a single decryption reveals the election result instead of decrypting individual votes. Also, any computation using the ciphertexts reveals no information about the individual votes and hence, the votes can't be manipulated. In contrast, [35] and [8] are both defined in terms of RSA moduli, and the number of messages that can be encrypted is of the same size as the modulus. When appropriate security levels are used, these allow far more messages than necessary for typical voting scenarios.

Castagnos and Laguillaumie [12] also presented numerical results using an implementation of their cryptosystem, which suggested that it has advantages over Pailler and Bresson *et al.* at the 112- and 128-bit security levels. However, the implementation was done using a general-purpose computer algebra system as opposed to a more specialized and optimized

implementation. In addition, two possible improvements were suggested, designed to allow larger messages without increasing the security level, and to speed up decryption via the Chinese Remainder Theorem. These improvements arise from using conductors that are prime powers and products of distinct primes, respectively, as opposed to primes. Exploring both these ideas was left as future work.

1.2 Contributions

In this thesis, we fully explore the efficiency of the cryptosystem of Castagnos and Laguillaumie [12]. Our first contribution is a complete description of the cryptosystem using conductors that are a power of a product of primes, thereby covering both the suggested improvements in [12]. We present a detailed benchmarking of the cryptosystem at the 128-, 192-, and 256-bit security levels, and compare its performance to both the Pailler [35] and the Bresson *et al.* [8] cryptosystems. Our implementation makes use of a state-of-the-art C implementation of class group arithmetic in imaginary quadratic orders due to Sayles [43]. We use both the original version of [12] where group elements are sampled from the entire group, as well as standard short exponent versions that also have provable security properties but under variations of the intractability assumptions that are restricted to short exponents, based on the results of Koshiba and Kurosawa [30]. The variations of Castagnos and Laguillaumie considered here offer performance improvements when using large exponents and large messages. When compared to the cryptosystems of Paillier [35] and Bresson *et al.* [8] at the same security levels, our results show that the basic version of Castagnos and Laguillaumie is the fastest at high security levels for small messages.

We presented this work in [18]. The first author of [18] implemented schemes by Paillier, Bresson *et al.* and Castagnos and Laguillaumie and ran experiments to find optimal runtimes for a given security level and message length. The author also presented improvements to the encryption schemes by Castagnos and Laguillaumie [12] by following their ideas and wrote

Sections 2–6 of [18] with assistance from the second and third authors of [18].

1.3 Organisation of Thesis

In Chapter 2, we introduce the mathematical background from number fields necessary to understand this thesis. Since the work of Castagnos and Laguillaumie is based on class groups of imaginary quadratic fields, we build our way to understanding these structures. In Chapter 3, we describe the Castagnos and Laguillaumie cryptosystem in detail. We present all the system variants, including the observations and conclusions presented by Castagnos and Laguillaumie in [12]. In Chapter 4, we present our novel contributions to the Castagnos and Laguillaumie cryptosystem. In Chapter 5, we present the results of our benchmarking experiments and draw comparative conclusions between the Paillier, Bresson *et al.* and Castagnos and Laguillaumie cryptosystems. Finally, we provide a summary of conclusions and present problems for future work in Chapter 6.

Chapter 2

Background

In this chapter we present the mathematical background necessary to understand our thesis. Primarily, we need to understand what a class group of an imaginary quadratic field is. Once we understand the concept of a class group of an imaginary quadratic field, we will see the computation of maps between such class groups. Most of the background material presented in this chapter are taken from [17] and [26]. For an introduction to elementary abstract algebra, the reader is referred to [19] and [34].

2.1 Imaginary Quadratic Fields

We begin with the concept of a quadratic irrational.

Definition 2.1.1. A quadratic irrational is an irrational number that is the of root of some quadratic equation with coefficients in $\mathbb{Q}$ that is irreducible over $\mathbb{Q}$. A quadratic irrational α can be represented as

$$\alpha = \frac{a + b\sqrt{c}}{d}$$

where $a, b, c \in \mathbb{Z}$, $b, c, d \neq 0$ and c is square-free.

One notes that the set of all quadratic irrationals is a subset of the set of complex numbers

$\mathbb{C}$. We now introduce the concept of a quadratic field.

Definition 2.1.2. Let $\alpha \in \mathbb{C}$ be a quadratic irrational. The field $\mathbb{K} = \mathbb{Q}(\alpha) = \{a + b\alpha : a, b \in \mathbb{Q}\}$ is an algebraic number field of degree 2 over the field of rational numbers $\mathbb{Q}$ and $\mathbb{K}$ is called a quadratic number field or simply, a quadratic field.

Since α is the root of a quadratic equation with rational coefficients, we can clear the denominators on both sides of the equation and α satisfies a quadratic equation $ax^2 + bx + c$ where $a, b, c \in \mathbb{Z}$ with $a \neq 0$. We then have,

$$\alpha = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Write $b^2 - 4ac = f^2 D$ where f and D are integers with $f > 0$ and D square-free. As shown in [26, page 77], since $\mathbb{Q}(\alpha) = \{x + y\alpha : x, y \in \mathbb{Q}\}$, one can write

$$\begin{aligned}
\mathbb{Q}(\alpha) &= \{x + y\alpha : x, y \in \mathbb{Q}\} \\
&= \left\{ x + y\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} : x, y \in \mathbb{Q}, \quad a, b \in \mathbb{Z} \right\} \\
&= \left\{ x + y\frac{-b \pm f\sqrt{D}}{2a} : x, y \in \mathbb{Q}, \quad a, b, f, D \in \mathbb{Z} \right\} \\
&= \left\{ x + \frac{-by}{2a} \pm \frac{yf}{2a}\sqrt{D} : x, y \in \mathbb{Q}, \quad a, b, f, D \in \mathbb{Z} \right\} \\
&= \left\{ a' + b'\sqrt{D} : a', b' \in \mathbb{Q} \right\} \\
&= \mathbb{Q}(\sqrt{D})
\end{aligned}$$

If $D < 0$, then $\mathbb{Q}(\sqrt{D})$ is called an *imaginary* quadratic field, and when $D > 0$, $\mathbb{Q}(\sqrt{D})$ is called a *real* quadratic field. From this point forward, we assume that D is negative. We next recall the concept of a quadratic integer.

Definition 2.1.3. A quadratic integer ξ is an element of $\mathbb{Q}(\sqrt{D})$ that is a root of a monic

quadratic polynomial whose coefficients are integers.

Proposition 2.1.1. *[26, Theorem 4.10] The quadratic integers of $\mathbb{Q}(\sqrt{D})$ are of the form*

$$\xi = \begin{cases} x + y\frac{1+\sqrt{D}}{2} & D \equiv 1 \pmod 4 \\ x + y\sqrt{D} & D \equiv 2, 3 \pmod 4 \end{cases}$$

for $x, y \in \mathbb{Z}$.

Let ξ_1 and ξ_2 be quadratic integers and let $[\xi_1, \xi_2]$ denote the $\mathbb{Z}$-module

$$\xi_1\mathbb{Z} + \xi_2\mathbb{Z} = \{\xi_1 x + \xi_2 y : x, y \in \mathbb{Z}\}$$

From Proposition 2.1.1, one can now note that $\alpha \in \mathbb{Q}(\sqrt{D})$ is a quadratic integer if and only if $\alpha \in [1, \omega]$ where

$$\omega = \begin{cases} \frac{1+\sqrt{D}}{2} & D \equiv 1 \pmod 4 \\ \sqrt{D} & D \equiv 2, 3 \pmod 4 \end{cases} \tag{2.1}$$

Example 2.1.1. Let $\mathbb{K} = \mathbb{Q}(\sqrt{-5})$. Then $D = -5 \equiv 3 \pmod 4$. The quadratic integers ξ of $\mathbb{K}$ take the form $\xi = x + y\sqrt{-5}$ where $x, y \in \mathbb{Z}$.

The set of all quadratic integers in $\mathbb{Q}(\sqrt{D})$ is a subring of $\mathbb{C}$ [16, page 47]. This ring structure is in fact called a *quadratic order*.

2.1.1 Orders in $\mathbb{K}$

Definition 2.1.4. [17, page 120] An order $\mathcal{O}$ in $\mathbb{K} = \mathbb{Q}(\sqrt{D})$ is a subset $\mathcal{O} \subset \mathbb{K}$ such that $\mathcal{O}$ is $\mathcal{O}$ is a subring of $\mathbb{K}$ containing 1 and is a free $\mathbb{Z}$-module of rank 2.

The quadratic order $\mathcal{O}$ that consists of all the quadratic integers in $\mathbb{Q}(\sqrt{D})$ is called the maximal order $\mathcal{O}_\mathbb{K}$ of $\mathbb{K} = \mathbb{Q}(\sqrt{D})$. All orders of $\mathbb{K}$ that are not $\mathcal{O}_\mathbb{K}$ are submodules of $\mathcal{O}_\mathbb{K}$ and are called non-maximal orders of $\mathbb{K}$.

Proposition 2.1.2. *[26, page 80] Let ω be a quadratic integer of $\mathbb{Q}(\sqrt{D})$ such that Equation (2.1) holds. The $\mathbb{Z}$-module with basis $[1, \omega]$ is the maximal order of $\mathbb{K} = \mathbb{Q}(\sqrt{D})$.*

Proposition 2.1.3. *[26, Theorem 4.17] If $\mathcal{O}$ is any order of $\mathbb{K}$, then $\mathcal{O} \subseteq \mathcal{O}_{\mathbb{K}}$ and $\mathcal{O} = [1, f\omega]$ for some $f \in \mathbb{Z}$.*

If $f = 1$, then $\mathcal{O} = \mathcal{O}_{\mathbb{K}}$. The integer f is called the *conductor* of $\mathcal{O}$. We now define an important invariant of a quadratic order.

Definition 2.1.5. [26, Definition 4.16] Let $\mathcal{O} = [\xi_1, \xi_2]$ denote any order of $\mathbb{Q}(\sqrt{D})$. The *discriminant* Δ of $\mathcal{O}$ is defined as

$$\Delta = \begin{vmatrix} \xi_1 & \xi_2 \\ \overline{\xi_1} & \overline{\xi_2} \end{vmatrix}^2 = (\xi_1\overline{\xi_2} - \overline{\xi_1}\xi_2)^2$$

where $\overline{\xi}$ denotes the complex conjugate of ξ. The discriminant of $\mathcal{O}$ is independent of the chosen $\mathbb{Z}$-basis.

By [26, Definition 4.16], the discriminant $\Delta_{\mathbb{K}}$ of the maximal order $\mathcal{O}_{\mathbb{K}} = [1, \omega]$ can be written as

$$\Delta_{\mathbb{K}} = \begin{cases} D & D \equiv 1 \pmod 4 \\ 4D & D \equiv 2, 3 \pmod 4 \end{cases} \tag{2.2}$$

The discriminant $\Delta_{\mathbb{K}}$ of the maximal order is an invariant of the quadratic field $\mathbb{K} = \mathbb{Q}(\sqrt{D})$ and is called the *fundamental discriminant* of $\mathbb{K}$.

Proposition 2.1.4. *[16, page 216] Let $\mathcal{O}$ denote an order of $\mathbb{Q}(\sqrt{D})$ with discriminant Δ. Then $\Delta = f^2\Delta_{\mathbb{K}}$ where $f \in \mathbb{N}$ is the conductor of $\mathcal{O}$ and f is the index of $\mathcal{O}$ in $\mathcal{O}_{\mathbb{K}}$ i.e., $f = [\mathcal{O}_{\mathbb{K}} : \mathcal{O}]$.*

For each positive integer f, there is a unique quadratic order in $\mathbb{K}$ of conductor f. Henceforth, we write $\Delta_f = f^2\Delta_{\mathbb{K}}$ and denote the order of discriminant Δ_f by $\mathcal{O}_{\Delta_f}$ and the maximal order by $\mathcal{O}_{\Delta_{\mathbb{K}}}$.

Example 2.1.2. Let $\mathbb{K} = \mathbb{Q}(\sqrt{-5})$. Since $D = -5 \equiv 3 \pmod 4$, $\omega = \sqrt{-5}$ from Equation (2.1). The maximal order of $\mathbb{K}$ is thus $\mathcal{O}_{\Delta_\mathbb{K}} = [1, \sqrt{-5}]$. The non-maximal order of conductor $f = 2$ is $\mathcal{O}_{\Delta_2} = [1, 2\sqrt{-5}]$.

2.1.2 Ideals in $\mathcal{O}$

In this section we will describe a particular structure of orders that is crucial to our understanding of the class group of an imaginary quadratic field.

Definition 2.1.6. [26, Definition 4.20] A non-zero integral ideal $\mathfrak{a}$ (*i.e.*, $\mathfrak{a} \neq 0$) of an order $\mathcal{O}$ is an additive subgroup of $\mathcal{O}$ such that $\xi\mathfrak{a} \subseteq \mathfrak{a}$ for every algebraic integer $\xi \in \mathcal{O}$.

From this point forward, we will only consider non-zero ideals in our discussion.

Definition 2.1.7. [26, Definition 5.1] An integral ideal $\mathfrak{a}$ of $\mathcal{O}$ is called primitive if $\mathfrak{a}$ cannot be written as $\mathfrak{a} = m\mathfrak{b}$ where $\mathfrak{b}$ is an ideal of $\mathcal{O}$, $m \in \mathbb{Z}$ and $|m| > 1$.

The ideals of $\mathcal{O}$ are $\mathcal{O}$-submodules of rank 2 [16, pages 125–126] *i.e.*, if $\mathfrak{a}$ is an ideal of $\mathcal{O}$, $\mathfrak{a}$ can be written as $\mathfrak{a} = (\alpha, \beta) = \{\xi_1\alpha + \xi_2\beta \mid \xi_1, \xi_2 \in \mathcal{O}\}$ where $\alpha, \beta \in \mathfrak{a}$ are generators of $\mathfrak{a}$.

Example 2.1.3. Let $\mathcal{O} = [1, \sqrt{-5}]$ be the maximal order of $\mathbb{Q}(\sqrt{-5})$ and let $\mathfrak{a} = \{2\xi_1 + \sqrt{-5}\xi_2 : \xi_1, \xi_2 \in \mathcal{O}\}$ be an additive subgroup of $\mathcal{O}$. Since an element ξ in $\mathcal{O}$ can be written as $\xi = x + y\sqrt{-5}$ for $x, y \in \mathbb{Z}$ from Proposition 2.1.1, we can write $\xi_1 = x_1 + y_1\sqrt{-5}$ and $\xi_2 = x_2 + y_2\sqrt{-5}$ with $x_1, x_2, y_1, y_2 \in \mathbb{Z}$. Let $\xi = x + y\sqrt{-5}$ in $\mathcal{O}$. Then

$$\begin{aligned}
\xi\alpha &= (x + \sqrt{-5}y)(2\xi_1 + \sqrt{-5}\xi_2) \\
&= (x + \sqrt{-5}y)(2(x_1 + \sqrt{-5}y_1) + \sqrt{-5}(x_2 + \sqrt{-5}y_2)) \\
&= (x + \sqrt{-5}y)(2x_1 + 2\sqrt{-5}y_1 + \sqrt{-5}x_2 - 5y_2) \\
&= 2x_1x + 2\sqrt{-5}y_1x + \sqrt{-5}x_2x - 5y_2x + 2\sqrt{-5}x_1y - 10y_1y - 5x_2y - 5\sqrt{-5}y_2y \\
&= 2((x_1x - 5y_1y) + \sqrt{-5}(y_1x + x_1y)) + \sqrt{-5}((x_2x - 5y_2y) + \sqrt{-5}(y_2x + x_2y)).
\end{aligned}$$

Since x, x_1, x_2, y, y_1, y_2 are integers, (x_1x-5y_1y), (y_1x+x_1y), (x_2x-5y_2y), (y_2x+x_2y) are also integers. Let $\xi_3 = (x_1x - 5y_1y) + \sqrt{-5}(y_1x + x_1y)$ and $\xi_4 = (x_2x - 5y_2y) + \sqrt{-5}(y_2x + x_2y)$. Then $\xi\alpha = 2\xi_3 + \sqrt{-5}\xi_4 \in \mathfrak{a}$ and thus, $\mathfrak{a}$ is an ideal of $\mathcal{O}$.

An ideal that has only one generator is called a principal ideal. If $\mathfrak{a}$ is a principal ideal generated by α in $\mathfrak{a}$, then we write $\mathfrak{a}$ as $\mathfrak{a} = (\alpha) = \{\xi\alpha : \xi \in \mathcal{O}\}$ [26, page 87]. We next describe a reduced ideal.

Definition 2.1.8. [26, Definition 5.4] Let $\mathcal{O}$ be an imaginary quadratic order and $\mathfrak{a}$ be an $\mathcal{O}$-ideal. Then $\mathfrak{a}$ is called reduced if $\mathfrak{a}$ is primitive and $\mathfrak{a}$ does not contain any non-zero element α such that $|\alpha| < N(\mathfrak{a})$.

In [26], Jacobson and Williams presented some simple conditions on the norm of $\mathfrak{a}$ that help distinguish a reduced ideal.

Proposition 2.1.5. *[26, Theorem 5.6] Let $\mathfrak{a}$ be a primitive ideal of $\mathcal{O}$. The ideal $\mathfrak{a}$ is reduced if $N(\mathfrak{a}) < \sqrt{|\Delta|}/2$.*

Definition 2.1.9. [26, Definition 4.27] Let $\mathfrak{a} = (\alpha_1, \beta_1)$ and $\mathfrak{b} = (\alpha_2, \beta_2)$ be ideals of $\mathcal{O}$. Then the product of $\mathfrak{a}$ and $\mathfrak{b}$ is defined as

$$\mathfrak{ab} = (\alpha_1\alpha_2, \alpha_1\beta_2, \beta_1\alpha_2, \beta_1\beta_2)$$

By [26, page 88], the product $\mathfrak{ab}$ is again an ideal of $\mathcal{O}$. The principal ideal $(1) = \mathcal{O}$ is the identity element with respect to ideal multiplication, since $\mathfrak{a} = \mathfrak{a}\mathcal{O} = \mathcal{O}\mathfrak{a}$. Shanks developed a technique, called NUCOMP, to compute the product of two ideals [45] [26, page 119]. This technique was improved by Sayles [42] which we use as part of our thesis. We also use Sayles' implementation of the ideal reduction algorithm [42, Algorithm 2.1]. The concept of principal ideals serves to define an equivalence relation on ideals.

Definition 2.1.10. [26, page 88] Two ideals $\mathfrak{a}$ and $\mathfrak{b}$ of $\mathcal{O}$ are equivalent if there exist $\alpha, \beta \in \mathcal{O}$ such that $(\alpha)\mathfrak{a} = (\beta)\mathfrak{b}$ and $\alpha\beta \neq 0$.

Definition 2.1.11. [26, Definition 4.33] Let $\mathfrak{a}$ be an ideal of $\mathcal{O}$. The norm of $\mathfrak{a}$, $N(\mathfrak{a})$, is defined to be the index $|\mathcal{O}/\mathfrak{a}|$.

Example 2.1.4. [2, Example 12.1.1] Let $\mathbb{K} = \mathbb{Q}(\sqrt{-5})$ and $\mathcal{O}_{\Delta_\mathbb{K}} = [1, \sqrt{-5}]$ be the maximal order of $\mathbb{K}$. Since $D = -5 \equiv 3 \pmod 4$, $\Delta_\mathbb{K} = 4 \cdot -5 = -20$. The $\mathcal{O}$-ideals $\mathfrak{a} = \left(2, 1 - \sqrt{-5}\right)$ and $\mathfrak{b} = (3, 1 + \sqrt{-5})$ are equivalent since

$$(3)\mathfrak{a} = (1 - \sqrt{-5})\mathfrak{b}$$

for principal ideals (3) and $(1 - \sqrt{-5})$ of $\mathcal{O}$.

If $\mathfrak{a}$ is an $\mathcal{O}$-ideal, then there exists an infinitude of ideals that are equivalent to $\mathfrak{a}$ [26, Section 5.1]. We shall denote the equivalence class of $\mathfrak{a}$ as $[\mathfrak{a}]$, where $[\mathfrak{a}]$ represents $\mathfrak{a}$ and all $\mathcal{O}$-ideals that are equivalent to $\mathfrak{a}$. From [26, page 88], we see that ideals are closed under multiplication, and by [16, page 121], ideal multiplication is associative, commutative, and has an identity which is the principal ideal $\mathcal{O}$. For ideals to form a group, we need inverses, and for this we define *fractional ideals*.

Definition 2.1.12. [26, Definition 7.3] A fractional ideal of $\mathcal{O}$ is a subset $\mathfrak{f}$ of $\mathbb{Q}(\sqrt{D})$ such that $\mathfrak{a} = d \cdot \mathfrak{f}$ is an integral ideal of $\mathcal{O}$ for some positive integer d.

From [9, Equation 8.8], a fractional ideal $\mathfrak{a}$ of an order $\mathcal{O}$ with discriminant Δ can be written as

$$\mathfrak{a} = q\left(a\mathbb{Z} + \frac{b + \sqrt{\Delta}}{2}\mathbb{Z}\right)$$

where q is a positive rational number, a and b are integers, $a > 0$ and $\Delta = b^2 - 4ac$ for some $c \in \mathbb{Z}$. Here, q and a are uniquely determined and b is determined $\pmod{2a}$. The integer b is chosen from the range $[-a, a]$ in order for $\mathfrak{a}$ to have a unique representation. In this case, $q(a, b, c)$ is called the standard representation for $\mathfrak{a}$ and the norm of $\mathfrak{a}$ is $N(\mathfrak{a}) = q^2 a$. If $q \in \mathbb{N}$, then $\mathfrak{a}$ is an integral ideal and if $q = 1$, then $\mathfrak{a}$ is an integral primitive ideal. In this case, we simply write the standard representation of $\mathfrak{a}$ as $\mathfrak{a} = (a, b)$ which uniquely describes $\mathfrak{a}$ since

the quantity c is determined by the discriminant Δ and the coefficients a, b. The norm of this ideal is then a. A fractional ideal $\mathfrak{a}$ of $\mathcal{O}$ is invertible if there is another fractional ideal $\mathfrak{b}$ such that $\mathfrak{ab} = \mathcal{O}$. All fractional ideals of the maximal order $\mathcal{O}_{\Delta_\mathbb{K}}$ are invertible but this is not true for all fractional ideals of non-maximal orders. The only fractional ideals of the non-maximal order $\mathcal{O}_{\Delta_f}$ that are invertible are those that are prime to the conductor f.

2.1.3 Ideals Prime to f

Definition 2.1.13. [24, Definition 2] Let $\mathcal{O}$ be any order and let $m \in \mathbb{N}$. A non-zero ideal $\mathfrak{a}$ of $\mathcal{O}$ is prime to m if $\mathfrak{a} + m\mathcal{O} = \mathcal{O}$.

We then have the following observation for invertible ideals of the non-maximal order $\mathcal{O}_{\Delta_f}$.

Proposition 2.1.6. *[24, Proposition 3] Let $\mathcal{O}_{\Delta_f}$ be a non-maximal order of conductor f and let $\mathfrak{a}$ be an ideal of $\mathcal{O}_{\Delta_f}$. Then*

- *$\mathfrak{a}$ is invertible if and only if $\mathfrak{a}$ is prime to the conductor f.*
- *$\mathfrak{a}$ is prime to the conductor f if and only if the norm of $\mathfrak{a}$ is prime to the conductor f.*

Thus, the set of invertible ideals of $\mathcal{O}_{\Delta_f}$ is comprised of exactly those fractional ideals that are prime to f and forms a group under ideal multiplication which we denote as $I(\mathcal{O}_{\Delta_f}, f)$. There exists a subgroup of $I(\mathcal{O}_{\Delta_f}, f)$ which is the group of principal ideals of $\mathcal{O}_{\Delta_f}$ prime to f and is denoted by $P(\mathcal{O}_{\Delta_f}, f)$ [24, Proposition 4]. Given an ideal $\mathfrak{a}$ in $\mathcal{O}_{\Delta_\mathbb{K}}$ prime to f, one can map $\mathfrak{a}$ to an ideal in $\mathcal{O}_{\Delta_f}$ prime to f and vice versa.

Proposition 2.1.7. *[17, Proposition 7.20] Let $\mathcal{O}_{\Delta_f}$ be the order of conductor f and $\mathcal{O}_{\Delta_\mathbb{K}}$ be the maximal order in an imaginary quadratic field $\mathbb{K}$. Then*

- *If $\mathfrak{a}$ is an $\mathcal{O}_{\Delta_\mathbb{K}}$-ideal prime to f, then $\mathfrak{a} \cap \mathcal{O}_{\Delta_f}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f of the same norm.*

- *If $\mathfrak{a}$ is an $\mathcal{O}_{\Delta_f}$-ideal prime to f, then $\mathfrak{a}\mathcal{O}_{\Delta_\mathbb{K}}$ is an $\mathcal{O}_{\Delta_\mathbb{K}}$-ideal prime to f of the same norm.*

- *The map $\varphi_f \colon I(\mathcal{O}_{\Delta_f}, f) \to I(\mathcal{O}_{\Delta_\mathbb{K}}, f)$ such that $\mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_\mathbb{K}}$ is an isomorphism.*

The map φ_f from Proposition 2.1.7 induces a surjection $\pi \colon C(\mathcal{O}_{\Delta_f}) \to C(\mathcal{O}_{\Delta_\mathbb{K}})$. We next present the map ψ used in the Castagnos and Laguillaumie encryption scheme.

Proposition 2.1.8. *[12, Lemma 3] Let p be a prime and let π denote the canonical surjection $\pi \colon C(\mathcal{O}_{\Delta_p}) \to C(\mathcal{O}_{\Delta_\mathbb{K}})$. The map $\psi \colon C(\mathcal{O}_{\Delta_\mathbb{K}}) \to C(\mathcal{O}_{\Delta_p})$ such that $h \mapsto h_l^p$, where $h_l \in \pi^{-1}(h)$ is an effective injective morphism.*

In [24], Hühnlein *et al.* presented algorithms that explicitly and efficiently compute the maps φ and φ^{-1} from Proposition 2.1.7. These algorithms play an important role in our understanding of this thesis. They require the following observation in the computation of these maps.

Proposition 2.1.9. *[17, Lemma 7.17] Let $\mathcal{O}$ be any order in any imaginary quadratic field. Given a non-zero integer f, every ideal class of $\mathcal{O}$ contains an ideal prime to f.*

In Algorithm 2.1.1, we present the algorithm from [24, Algorithm 8] that takes a primitive $\mathcal{O}$-ideal $\mathfrak{a} = (a, b)$ and a non-zero integer f as its inputs and returns an ideal $\mathfrak{b} = (a', b')$ equivalent to $\mathfrak{a}$ that is prime to f. Next, Algorithm 2.1.2 computes the map φ from Proposition 2.1.7 and Algorithm 2.1.3 computes φ^{-1} from Proposition 2.1.7.

2.2 Ideal Class Group

In this section, we define the ideal class group of a quadratic order in a quadratic field.

Definition 2.2.1. [26, Definition 7.1] Let $\mathcal{O} = \mathcal{O}_{\Delta_f}$ be an order of discriminant Δ_f. The ideal class group $C(\mathcal{O})$ of $\mathcal{O}$ is defined as

Algorithm 2.1.1 FindIdealPrimeTo – [24, Algorithm 8]

Input: A primitive ideal $\mathfrak{a} = (a, b)$ of $\mathcal{O}$ of discriminant Δ and a non-zero integer f

Output: A primitive $\mathfrak{b} = (a', b')$ equivalent to $\mathfrak{a}$ such that $\gcd(a', f) = 1$

1. **if** $\gcd(a, f) > 1$ **then**
2. Set $c \leftarrow (b^2 - \Delta)/4a$
3. **if** $\gcd(c, f) > 1$ **then**
4. Set $a' \leftarrow a + b + c$
5. Set $b' \leftarrow -b - 2a$
6. **else**
7. Set $a' \leftarrow c$
8. Set $b' \leftarrow -b$
9. **return** (a', b')
10. **end if**
11. **else**
12. **return** (a, b)
13. **end if**

Algorithm 2.1.2 GoToNonMaxOrder – [24, Algorithm 9]

Input: A primitive ideal $\mathfrak{a}$ of $\mathcal{O}_{\Delta_{\mathbb{K}}}$ and a positive integer f

Output: A primitive ideal $\mathfrak{b} = (a, b)$ of $\mathcal{O}_{\Delta_f}$ such that $\mathfrak{b} = \varphi(\alpha\mathfrak{a}) = (\alpha\mathfrak{a}) \cap \mathcal{O}_{\Delta_f}$ where $\alpha\mathfrak{a}$ is an ideal prime to f

1. $(a, b) \leftarrow$ FindIdealPrimeTo$(\mathfrak{a}, f)$
2. $b \leftarrow bf \pmod{2a}$
3. **return** (a, b)

Algorithm 2.1.3 GoToMaxOrder – [24, Algorithm 10]

Input: A primitive ideal $\mathfrak{a}$ of $\mathcal{O}_{\Delta_f}$ and a positive integer f

Output: A primitive ideal $\mathfrak{b} = (a, b)$ of $\mathcal{O}_{\Delta_\mathbb{K}}$ such that $\mathfrak{b} = \varphi^{-1}(\alpha\mathfrak{a}) = (\alpha\mathfrak{a})\mathcal{O}_{\Delta_\mathbb{K}}$ where $\alpha\mathfrak{a}$ is an ideal prime to f

1. $(a, b) \leftarrow \text{FindIdealPrimeTo}(\mathfrak{a}, f)$
2. Set $b_0 \leftarrow \Delta_f \pmod 2$
3. Solve $1 = \mu f + \lambda a$ for $\mu, \lambda \in \mathbb{Z}$
4. $b \leftarrow b\mu + ab_0\lambda \pmod{2a}$
5. **return** (a, b)

$$C(\mathcal{O}) = I(\mathcal{O}_{\Delta_f}, f)/P(\mathcal{O}_{\Delta_f}, f)$$

where $I(\mathcal{O}_{\Delta_f}, f)$ is the group of invertible fractional ideals of $\mathcal{O}_{\Delta_f}$ that are prime to f and $P(\mathcal{O}_{\Delta_f}, f)$ the subgroup of $I(\mathcal{O}_{\Delta_f}, f)$ consisting of principal ideals.

The class group is a finite abelian group as shown in [26, Section 7.1]. The elements of the class group $C(\mathcal{O})$ are ideal classes $[\mathfrak{a}]$ where $\mathfrak{a}$ is invertible. Since the class group is defined under multiplication, we next see how to efficiently multiply two ideal classes $[\mathfrak{a}]$ and $[\mathfrak{b}]$ where $\mathfrak{a}, \mathfrak{b} \in \mathcal{O}$. Before that, we will need the definition of a *reduced ideal*. An ideal class $[\mathfrak{a}]$ contains an infinitude of ideals that are equivalent to $\mathfrak{a}$ but one requires a set of representatives that allows one to uniquely represent an equivalence class of $C(\mathcal{O})$. The elements of class groups of an imaginary quadratic field can be uniquely represented by reduced ideals.

Proposition 2.2.1. *[9, Section 9.3.2, Theorem 5.7.7] Let $\mathcal{O}$ be an order of discriminant $\Delta < 0$. Then every equivalence class of $C(\mathcal{O})$ contains exactly one reduced ideal.*

2.2.1 A Particular Subgroup of $C(\mathcal{O}_{\Delta_f})$

The map φ_f^{-1} from Proposition 2.1.7 induces a surjection $\bar{\varphi}_f \colon C(\mathcal{O}_{\Delta_f}) \to C(\mathcal{O}_{\Delta_\mathbb{K}})$ and in [17], Cox computed the order of $\ker(\bar{\varphi}_f)$ as follows.

Theorem 2.2.1. *[17, Proposition 7.22 and Theorem 7.24] Let $\Delta_{\mathbb{K}}$ be a fundamental negative discriminant with $\Delta_{\mathbb{K}} \neq -3, -4$ and f be a positive integer. Then the order of $\ker \bar{\varphi}_f$ is given by*

$$f \prod_{p|f} \left(1 - \left(\frac{\Delta_{\mathbb{K}}}{p}\right) \frac{1}{p}\right)$$

where the product runs over all primes p that divide f and $(\Delta_{\mathbb{K}}/p)$ denotes the Kronecker symbol ([26, Definition 1.2]).

In Chapter 1, we mention that the Castagnos and Laguillaumie cryptosystem is based on the difficulty of solving the discrete logarithm problem in the class group of a maximal order whose order is unknown. We will now look at some results used by Castagnos and Laguillaumie to set up their cryptosystem in the class group setting.

2.2.2 Cohen-Lenstra Heuristics

We will now look at the ideal class group in more detail, presenting some important heuristic results due to Cohen and Lenstra. In the previous section, we defined the class group $C(\mathcal{O})$. The cardinality of the class group $C(\mathcal{O}_\Delta)$ is called the class number and is denoted as $h(\mathcal{O}_\Delta)$. The *odd part* of the class group $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ is the subgroup of ideal classes whose orders are odd and the *even part* of the class group is the subgroup of ideal classes whose orders are even. Cohen and Lenstra presented several heuristic results in [15] regarding the class group of an imaginary quadratic field $\mathbb{K}$. We present some of these results below necessary for our understanding of this thesis:

1. The heuristic probability that the odd part of the class group is cyclic is 97.757%.
2. The heuristic probability that an odd prime p divides the class number $h(\Delta_{\mathbb{K}})$ is approximately $1/p + 1/p^2$.

In [14, page 296], Cohen also stated that if $\Delta < -4$, then,

$$h(\Delta_{\mathbb{K}}) < \frac{1}{\pi} \sqrt{|\Delta_{\mathbb{K}}|} \log |\Delta_{\mathbb{K}}|. \tag{2.3}$$

2.2.3 Discrete Logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$

We now discuss how to compute the discrete logarithm in the class group $C(\mathcal{O}_{\Delta_\mathbb{K}})$. The discrete logarithm problem in a group G can be stated as follows.

Definition 2.2.2. Let G be a cyclic group of order n and g be a generator of G. Let $x \xleftarrow{\$} \{0, \ldots, n-1\}$ denote the operation of selecting a random element x from the set $\{0, \ldots, n-1\}$. Given g and g^x but not x where $x \xleftarrow{\$} \{0, \ldots, n-1\}$, the discrete logarithm problem consists of computing x.

Since the class group is a finite abelian group, one can use generic group algorithms to compute the discrete logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ where $\mathcal{O}_{\Delta_\mathbb{K}}$ is the maximal order of discriminant $\Delta_\mathbb{K}$. However, to compute discrete logarithms in $C(\mathcal{O}_{\Delta_\mathbb{K}})$, generic algorithms such as the Baby-Step-Giant-Step algorithm require at least an upper bound on the class number $h(\Delta_\mathbb{K})$. The algorithm then runs in at most $\sqrt{n}$ steps where n is the order of the group or an upper bound on the order of the group [44]. From equation (2.3), we see that $h(\Delta_\mathbb{K}) \approx \sqrt{\Delta_\mathbb{K}}$, giving us an idea on how big the fundamental discriminant $\Delta_\mathbb{K}$ has to be to make the generic algorithms computationally infeasible in $C(\mathcal{O}_{\Delta_\mathbb{K}})$. In [22], Hafner and McCurley presented an algorithm to compute the discrete logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ with a sub-exponential complexity of $\exp(\sqrt{\ln|\Delta_\mathbb{K}| \ln\ln|\Delta_\mathbb{K}|})^{\sqrt{2}+o(1)}$. In [6], this algorithm was improved to have a conjectured sub-exponential complexity $\exp(\sqrt{\ln|\Delta_\mathbb{K}| \ln\ln|\Delta_\mathbb{K}|})^{1+o(1)}$.

In [6], Biasse *et al.* presented methods on how to choose the fundamental discriminant and estimated the sizes of $\Delta_\mathbb{K}$ to make the index-calculus algorithm infeasible at any given security level. In [23], Hamdy and Möller mention some considerations that one has to take during the selection of $\Delta_\mathbb{K}$ that make the computation of discrete logarithms in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ as hard as in finite fields. Essentially, they recommended using a fundamental discriminant $\Delta_\mathbb{K}$ and minimising the size of the 2-Sylow subgroup of the class group. We will now see how to do this.

2.2.4 The 2-Sylow Subgroup

To understand how to minimise the size of the 2-Sylow subgroup, we will first present the definition of a 2-Sylow subgroup and then talk about the p-rank of the class group.

Definition 2.2.3. For a prime p, the p-Sylow subgroup of a finite group G is defined as a subgroup of G of order p^k where p^k is the highest power of p dividing the order of G.

Following the fundamental theorem for finite Abelian groups, a class group $C(\mathcal{O})$ can be canonically decomposed into a direct product of its cyclic subgroups,

$$C(\mathcal{O}) \simeq C(m_1) \times \cdots \times C(m_s)$$

where $m_1, \cdots, m_s$ are positive integers such that $m_1 \geq 1$, $m_j \mid m_{j-1}$ for $2 \leq j \leq s$, and $C(m_i)$ denotes the cyclic group of order m_i for $1 \leq i \leq s$. For a prime p, the p-rank of the class group is defined as the number of m_i that are divisible by p.

Example 2.2.1. Let $D = -1 \cdot 7 \cdot 29 \cdot 17$ and $\mathbb{K} = \mathbb{Q}(\sqrt{D})$. Since $D \equiv 1 \pmod 4$, $\Delta_{\mathbb{K}} = D$ and using `quadclassunit()` from PARI/GP [36] we see,

$$C(\mathcal{O}_{\Delta_{\mathbb{K}}}) \simeq C(6) \times C(2)$$

The function also returns the class number $h(\Delta_{\mathbb{K}})$ which is 12 in this case. Since 6 and 2 are both divisible by 2, the 2-rank of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ is 2 and the order of the 2-Sylow subgroup is 4. Also, the odd part of the class group in this example is $C(3)$ whereas the even part is $C(2) \times C(2)$.

The divisors of the discriminant of an order $\mathcal{O}$ provide useful information about the 2-Sylow subgroup of the class group $C(\mathcal{O})$. If k is the number of distinct prime divisors of Δ, then $C(\mathcal{O}_\Delta)$ has 2-rank $k-1$, so 2^{k-1} divides $h(\mathcal{O}_\Delta)$. If the prime divisors of the fundamental discriminant $\Delta_{\mathbb{K}}$ satisfy some Legendre symbol conditions, then one can show that 2^{k-1} is indeed the highest power of 2 that divides the size of the 2-Sylow subgroup. If $k = 2$, *i.e.*,

$\Delta_{\mathbb{K}}$ is a product of two primes p and q, then from [28, Proposition 0], we see that the size of the 2-Sylow subgroup is 2 $(= 2^{2-1})$ if

$$(p/q) = (q/p) = -1 \tag{2.4}$$

If $\Delta_K = -q_1 q_2 \cdots q_k$ and $k \geq 1$, then from [7], we see that the size of the 2-Sylow subgroup is 2^{k-1} if

$$(q_i/q_j) = (q_j/q_i) = 1 \text{ and } (q_i/q_k) = (q_k/q_i) = -1 \text{ for } 1 \leq i, j < k \text{ and } i \neq j \tag{2.5}$$

Note that Equation (2.5) includes Equation (2.4) for $k = 2$. If $k = 1$, the 2-Sylow subgroup is trivial *i.e.*, the class number $h(\Delta_{\mathbb{K}})$ is odd.

Chapter 3

Castagnos and Laguillaumie Encryption Scheme

3.1 Introduction

A linearly homomorphic encryption scheme is one where linear combinations of ciphertexts can be computed such that the decryption of this linear combination produces a plaintext that is identical to the same linear combination of the individual decryptions. Formally speaking, let the plaintext be a vector space over some field. Then linearly homomorphic encryption under any key is a linear map on the vector space of plaintexts. A linearly homomorphic encryption scheme consists of the following components:

1. An algorithm KeyGen that takes as input a security parameter λ and generates a user's public/private key pair $(pk, sk) = \mathsf{KeyGen}(\lambda)$.

2. An (randomized) algorithm $\mathsf{Encrypt}$ that takes as input a plaintext m and a public key pk and outputs a ciphertext $c = \mathsf{Encrypt}(m, pk)$.

3. An algorithm $\mathsf{Decrypt}$ that takes as input a ciphertext c that is the encryption of a plaintext m under pk as well as the corresponding private key sk and outputs $m =$

$\mathsf{Decrypt}(c, sk)$ of c.

4. Homomorphic properties:

   (a) $\mathsf{Encrypt}(m_1 + m_2) = \mathsf{Encrypt}(m_1) \cdot \mathsf{Encrypt}(m_2)$ for all plaintexts m_1 and m_2

   (b) $\mathsf{Encrypt}(\alpha m) = \mathsf{Encrypt}(m)^\alpha$ for all plaintexts m and scalars α from the field of scalars over which the vector space of plaintexts is defined.

Many linearly homomorphic encryption schemes have been proposed over the years with some of the popular ones being [35] and [8]. However, until 2015, none of the linearly homomorphic schemes were based solely on the discrete logarithm (abbreviated DL) problem. In 2015, Castagnos and Laguillaumie solved this thirty year old open problem by presenting a scheme based on a DL related assumption, the decisional Diffie-Hellman problem. We will now describe these problems one by one which can be found in [31, Chapter 3]. Before we describe the decisional Diffie-Hellman problem, we describe the Diffie-Hellman problem.

Definition 3.1.1. Let G be a cyclic group of order n and g be a generator of G. Let $x, y \xleftarrow{\$} \{0, \ldots, n-1\}$. Given g, g^x and g^y but not x and y, the (computational) Diffie-Hellman problem consists of computing g^{xy}.

Definition 3.1.2. Let G be a cyclic group of order n and g be a generator of G. Let $x, y \xleftarrow{\$} \{0, \ldots, n-1\}$. Given g, g^x, g^y but not x and y, the decisional Diffie-Hellman problem consists of deciding whether $g^{xy} = g^z$ where $z \xleftarrow{\$} \{0, \ldots, n-1\}$.

In this chapter, we will first describe the linearly homomorphic encryption scheme presented by Castagnos and Laguillaumie in [11] and follow it up with a discussion on its security features. Later on, we will look at a second encryption scheme proposed by Castagnos and Laguillaumie and then discuss some of the improvements proposed by Castagnos and Laguillaumie to extend their schemes.

3.2 Generic Castagnos and Laguillaumie Construction

The linearly homomorphic encryption scheme presented by Castagnos and Laguillaumie in [12] is based on the hardness of the decisional Diffie-Hellman (DDH) problem in certain groups $\mathcal{G}$ that contain a subgroup $\mathcal{F}$ where solving the DL problem is easy. Castagnos and Laguillaumie call such a setting a DDH group with an easy DL subgroup and instantiate an example of one such group-subgroup pair as the class group of a non-maximal imaginary quadratic order whose conductor f is a prime p, paired with a subgroup of order f. We begin our discussion of the Castagnos and Laguillaumie cryptosystem by presenting the definition of a *DDH group with an easy DL subgroup* as stated by Castagnos and Laguillaumie.

Definition 3.2.1. [12, Definition 1] A DDH group with an easy DL subgroup is a pair of algorithms Gen and Solve. The Gen algorithm takes as input two parameters λ and μ and outputs a tuple $(B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F})$. Here, $\mathcal{G}$ is a finite cyclic group generated by $\mathfrak{g}$ of unknown order, $\mathcal{F}$ is a cyclic subgroup of $\mathcal{G}$ of order f and generated by $\mathfrak{f}$, B is an upper bound on the order of the group $\mathcal{G}/\mathcal{F}$, where the order is a λ-bit integer, and f is a μ-bit integer. Solve is an efficient algorithm that computes the discrete logarithm in $\mathcal{F}$. The DDH problem in $\mathcal{G}$ is assumed to be hard even with access to the Solve algorithm while computing the DL problem in $\mathcal{F}$ is assumed to be easy.

In slight abuse of terminology, we will refer to $\mathcal{G}$ as a DDH group and $\mathcal{F}$ an easy DL subgroup of $\mathcal{G}$, with an implicit assumption of the associated Gen and Solve algorithms. The definition talks about two problems, namely the Decisional Diffie-Hellman (DDH) problem and the Discrete Logarithm (DL) problem which we are now going to introduce in the context of the group $\mathcal{G}$ described in Definition 3.2.1.

Definition 3.2.2. Let $\mathcal{G}$ be a DDH group of order n with an easy DL subgroup $\mathcal{F}$ and let $\mathfrak{g}$ be a generator of $\mathcal{G}$. Let x, y and z be random integers from the set $\{0, \ldots, n-1\}$. Given $(\mathfrak{g}, \mathfrak{g}^x, \mathfrak{g}^y, \mathfrak{g}^z)$ and access to the Solve algorithm, the Decisional Diffie Hellman Problem consists of deciding whether $\mathfrak{g}^{xy} = \mathfrak{g}^z$.

Castagnos and Laguillaumie presented a generic linearly homomorphic scheme that uses the concept of a group $\mathcal{G}$ with a subgroup $\mathcal{F}$ where the DDH problem is hard to solve in $\mathcal{G}$ and the DL problem is easy to solve in $\mathcal{F}$. In the following section, we discuss this generic construction from [12].

Using the idea of a hard DDH group $\mathcal{G}$ with an easy DL subgroup $\mathcal{F}$, Castagnos and Laguillaumie proposed a generic linearly homomorphic encryption scheme that is based on an Elgamal encryption [20] style. Algorithm 3.2.1 presents the key generation algorithm (abbreviated KeyGen) presented by Castagnos and Laguillaumie in [12]. KeyGen takes two integers λ and μ as its inputs and generates the public and secret keys of the system. It does this by first calling the Gen subroutine which returns the tuple $(B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F})$ as its output. The description of each entity of the tuple is described in Definition 3.2.1. KeyGen then computes a secret key x from the set $\{0, \ldots, Bf-1\}$ where $\xleftarrow{\$}$ denotes sampling a random element of a particular set, in this case, the set $\{0, \ldots, Bf-1\}$. For the public key, a random element $\mathfrak{h}$ of $\mathcal{G}$ is computed as $\mathfrak{g}^x$ and the tuple $(B, f, \mathfrak{f}, \mathfrak{g}, \mathfrak{h})$ is set as the public key.

Algorithm 3.2.1 KeyGen

Input: λ, μ

Output: Public key pk, secret key sk

1. $(B, f, \mathfrak{g}, \mathfrak{f}) \leftarrow \mathsf{Gen}(\lambda, \mu)$
2. Compute $x \xleftarrow{\$} \{0, \ldots, Bf-1\}$
3. Set $\mathfrak{h} \leftarrow \mathfrak{g}^x$
4. Set $pk \leftarrow (B, f, \mathfrak{g}, \mathfrak{h}, \mathfrak{f})$
5. Set $sk \leftarrow x$
6. Return (pk, sk)

The encryption algorithm (abbreviated Encrypt) takes the message m to be encrypted where $m \in (\mathbb{Z}/f\mathbb{Z})^*$ and the public key pk as inputs and outputs a pair of ciphertexts (c_1, c_2). The algorithm first generates an exponent r uniformly at random from the set $\{0, \ldots, Bf-1\}$ and computes the first ciphertext c_1 as $\mathfrak{g}^r$ where $\mathfrak{g}$ is the generator of the

group $\mathcal{G}$ from the public key tuple. The message m is then encoded in the exponent of $\mathfrak{f}$ and uniformly distributed in the group $\mathcal{G}$ by multiplying $\mathfrak{f}^m$ with $\mathfrak{h}^r$. Finally, the algorithm returns the pair of ciphertexts c_1 and c_2 as output. Algorithm 3.2.2 describes this encryption algorithm.

Algorithm 3.2.2 Encrypt

Input: pk, message m

Output: Ciphertext (c_1, c_2)

1. Pick $r \xleftarrow{\$} \{0, \ \ldots, \ Bf - 1\}$
2. Compute $c_1 \leftarrow \mathfrak{g}^r$
3. Compute $c_2 \leftarrow \mathfrak{f}^m \mathfrak{h}^r$
4. Return (c_1, c_2)

The decryption algorithm (abrreviated Decrypt) takes as inputs the ciphertexts (c_1, c_2), the public key pk and the secret key sk and produces the plaintext m as its output. It uses the secret key x to compute c_2/c_1^x which results in $\mathfrak{f}^m$. The algorithm then calls a subroutine Solve using the public key information and $\mathfrak{f}^m$ as its inputs. Solve takes the group $\mathcal{F}$ parameters and deterministically computes the discrete logarithm in $\mathcal{F}$ in polynomial time to produce the message m as its output. The details of the Gen and Solve algorithms however, depend on the specific instantiation of the groups $\mathcal{G}$ and $\mathcal{F}$. Algorithm 3.2.3 describes this decryption algorithm.

Algorithm 3.2.3 Decrypt

Input: $pk, sk, (c_1, c_2)$

Output: Message m

1. Compute $\mathfrak{m} \leftarrow c_2/c_1^x$
2. $m \leftarrow \mathsf{Solve}(\mathfrak{m}, f)$
3. Return m

Castagnos and Laguillaumie also presented two algorithms, EvalSum and EvalScal, that establish the linear homomorphic properties of the generic scheme. The EvalSum algorithm takes two pairs of ciphertexts (c_1, c_2), (c_1', c_2') and the public key pk as inputs and produces

a pair of ciphertexts C_1, C_2 as its output. Like Encrypt, it computes a random exponent r from the set $\{0, \ldots, Bf-1\}$ and then computes $\mathfrak{h}^r$ in $\mathcal{G}$. Next, it pair-wise computes the products c_1c_1' and c_2c_2' which are uniformly distributed in $\mathcal{G}$ throught multiplying with $\mathfrak{h}^r$. Thus, like Encrypt the EvalSum algorithm terminates by correctly computing a pair of ciphertexts (C_1, C_2) where $C_1 = c_1c_1'\mathfrak{g}^r$ and $C_2 = c_2c_2'\mathfrak{h}^r$. The ciphertext C_1 is a correct ciphertext as $c_1c_1' = \mathfrak{g}^r\mathfrak{g}^{r'} = \mathfrak{g}^{r+r'}$ and $r+r'$ is just another random element in $\{0, \cdots Bf-1\}$. The ciphertext C_2 is a correct ciphertext as $c_2c_2' = \mathfrak{f}^m\mathfrak{h}^r\mathfrak{f}^{m'}\mathfrak{h}^{r'} = \mathfrak{f}^{m+m'}\mathfrak{h}^{r+r'}$ and is the encryption of the plaintext $m + m' \pmod f$ and $r + r'$ is just another random element in $\{0, \ldots, Bf-1\}$. Thus, when (C_1, C_2) is passed as an input to the decryption algorithm, it correctly decrypts the ciphertext product $\mathfrak{f}^m\mathfrak{f}^{m'}$ to produce the plaintext sum $(m_1 + m_2) \pmod f$, thus establishing the first linear homomorphic property. Algorithm 3.2.4 describes this homomorphic addition algorithm.

Algorithm 3.2.4 EvalSum

Input: pk, (c_1, c_2), (c_1', c_2')

Output: (C_1, C_2) such that $\mathsf{Decrypt}(sk, (C_1, C_2)) = m + m'$

1. Compute $c_1'' \leftarrow c_1c_1'$, $c_2'' \leftarrow c_2c_2'$
2. Pick $r \xleftarrow{\$} \{0, \ldots, Bf-1\}$
3. Return $(c_1''\mathfrak{g}^r, c_2''\mathfrak{h}^r)$

While EvalSum verifies the additive homomorphic property of the cryptosystem, the EvalScal algorithm verifies the scalar multiplication homomorphic property of the cryptosystem. It takes three inputs, the public key pk, a pair of ciphertexts c_1, c_2 and a random positive integer α. It then computes a random exponent r from the set $\{0, \ldots, Bf-1\}$. The algorithm then computes c_1^α and c_2^α to produce ciphertexts (C_1, C_2) such that $C_1 = c_1^\alpha\mathfrak{g}^r = \mathfrak{g}^{r'\alpha}\mathfrak{g}^r$ and $C_2 = c_2^\alpha\mathfrak{h}^r = \mathfrak{f}^{m\alpha}\mathfrak{h}^{r'\alpha}\mathfrak{h}^r$ where it is easy to see that $r+r'\alpha$ is just another random element from the set $\{0, \ldots, Bf-1\}$. Thus C_1 and C_2 are correct ciphertexts as they have a random representation in the group $\mathcal{G}$ and is distributed uniformly in $\mathcal{G}$. Since EvalScal produces correct ciphertexts, the decryption algorithm correctly computes $m \cdot \alpha \pmod f$, thus estab-

lishing the scalar multiplication homomorphic property of the cryptosystem. Algorithm 3.2.5 describes this homomorphic scalar multiplication algorithm.

Algorithm 3.2.5 EvalScal

Input: pk, α, (c_1, c_2)

Output: (C_1, C_2) such that $\mathsf{Decrypt}(sk, (C_1, C_2)) = \alpha m$

1. Compute $c_1' \leftarrow c_1^{\alpha}$, $c_2' \leftarrow c_2^{\alpha}$
2. Pick $r \xleftarrow{\$} \{0, \ldots, Bf - 1\}$
3. Return $(c_1' \mathfrak{g}^r, c_2' \mathfrak{h}^r)$

In the following section, we are going to see the explicit instantiation of the groups $\mathcal{G}$ and $\mathcal{F}$ in the class group of a non-maximal quadratic order of an imaginary quadratic field $\mathbb{K}$. We will also present explicit descriptions of the Gen and Solve algorithms in this class group setting.

3.3 Instantiation in Class Groups

As mentioned before in Chapter 1, Castagnos and Laguillaumie instantiated the hard DDH group with an easy DL subgroup in the class group of a non-maximal imaginary quadratic order. They showed the existence of groups $\mathcal{G}$ and $\mathcal{F}$ with the following observation.

Theorem 3.3.1. *[12, Page 10] Let $\Delta_{\mathbb{K}}$ be a fundamental square-free negative discriminant of the form $\Delta_{\mathbb{K}} = -pq$ where p is an odd prime and q a positive integer prime to p such that $q > 4p$. Let (p^2, p) be an ideal of $\mathcal{O}_{\Delta_p}$, the order of discriminant $\Delta_p = p^2\Delta_{\mathbb{K}}$. Let $\mathfrak{f}$ denote the class of (p^2, p) in $C(\mathcal{O}_{\Delta_p})$. Then for $m \in \{1 \cdots p-1\}$, $\mathfrak{f}^m = [p^2, L(m)p]$ and this form is reduced (and hence, unique). Here, $L(m)$ is the odd integer in $[-p, p]$ such that $mL(m) \equiv 1 \pmod{p}$. Also, $\mathfrak{f}$ is a generator of the subgroup of order p in $C(\mathcal{O}_{\Delta_p})$.*

Thus, Castagnos and Laguillaumie use a non-maximal imaginary quadratic order of prime conductor $f = p$ in their instantiation of the generic LHE scheme. In Algorithm 3.2.1, we saw that KeyGen calls a subroutine Gen that computes the group parameters of the scheme.

We now look at the Gen algorithm presented by Castagnos and Laguillaumie that generates group parameters in the class group $C(\mathcal{O}_{\Delta_f})$ of a non-maximal imaginary quadratic order $\mathcal{O}_{\Delta_f}$. The Gen subroutine generates two random primes p and q and sets $\Delta_{\mathbb{K}} = -pq$ such that $\Delta_{\mathbb{K}} \equiv 1 \pmod 4$. Next, it sets the conductor f to the prime p, computes $\Delta_f = f^2\Delta_{\mathbb{K}}$ and sets $\mathfrak{f}$ to the ideal (f^2, f) in $C(\mathcal{O}_{\Delta_f})$. Using [24, Subsection 3.1], Gen then constructs an ideal $\mathfrak{r}$ of $\mathcal{O}_{\Delta_{\mathbb{K}}}$ with norm r where r is a prime and $(\Delta_{\mathbb{K}}/r) = 1$. From Subsection 2.2.4, the ideal class $[\mathfrak{r}^2]$ lies in the odd part of the class group as $r \nmid \Delta_{\mathbb{K}}$. Thus, $[\mathfrak{r}^2]$ has odd order that divides s where s denotes the size of the odd part of the class group $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. This is because the odd part of the class group is cyclic with very high probability due to the first Cohen-Lenstra heuristic mentioned in Subsection 2.2.2. Knowing f and using Algorithm 2.1.2, Gen lifts $[\mathfrak{r}^2]$ to $C(\mathcal{O}_{\Delta_f})$ and since ψ is injective from Proposition 2.1.8, one gets a class of $C(\mathcal{O}_{\Delta_f})$ of order s. Castagnos and Laguillaumie recommend choosing a conductor p of length at least 80 bits so that $\gcd(p, s) = 1$ with high probability from the second Cohen-Lenstra heuristic given in Section 2.2. Thus, multiplying this class with $\mathfrak{f}^k$ gives us an element $\mathfrak{g}$ of order ps. Note that the element $\mathfrak{g}$ is still a square of $C(\mathcal{O}_{\Delta_f})$ as it is a product of two squares: $\mathfrak{g}$ is a square by construction, $\mathfrak{f} = (\mathfrak{f}^{2^{-1} \pmod f})^2$ and the inverse of 2 modulo f exists because f is odd. Castagnos and Laguillaumie show in [12, Appendix C] that the statistical distance of the uniform distribution on $\mathcal{G}$ to the uniform distribution $\{\mathfrak{g}^r, r \xleftarrow{\$} \{0, \ldots, B-1\}\}$, where B is an upper bound on $|\mathcal{G}|$, is upper bounded by $|G|/4B$. Using Equation (2.3), they set B to $f \cdot \left\lceil \frac{\log(|\Delta_{\mathbb{K}}|) \cdot \sqrt{|\Delta_{\mathbb{K}}|}}{4\pi} \right\rceil$, so that random powers $\mathfrak{g}^r$ with $0 \leq r \leq Bf - 1$ can be assumed to be statistically indistinguishable from the uniform distribution on $\mathcal{G}$. Algorithm 3.3.1 describes the pseudocode of the Gen algorithm.

As mentioned in Definition 3.2.1, the Solve algorithm is used to compute discrete logarithms in $\mathcal{F}$. We now present the Solve algorithm given by Castagnos and Laguillaumie that solves the discrete logarithm problem in $\mathcal{F} \subset C(\mathcal{O}_{\Delta_f})$. We have seen that Encrypt encrypts a message $m \in (\mathbb{Z}/f\mathbb{Z})^\star$, to produce two ciphertexts c_1 and c_2, which are ideal classes in $C(\mathcal{O}_{\Delta_f})$. We have also seen that Decrypt takes these ciphertexts and computes $\mathfrak{f}^m$ which is

Algorithm 3.3.1 Gen

Input: $\lambda,\ \mu$ with $\lambda \geq \mu + 2$.

Output: $B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F}$

1. Pick random integers p and q such that p is a μ-bit prime, q is a $(2\lambda - \mu)$-bit prime, $pq \equiv 3 \pmod 4$ and $(p/q) = (q/p) = -1$
2. Set $\Delta_{\mathbb{K}} \leftarrow -pq$
3. Set $f \leftarrow p$
4. Set $\Delta_f \leftarrow f^2 \Delta_{\mathbb{K}}$
5. Set $\mathfrak{f} \leftarrow [(f^2, f)]$ in $C(\mathcal{O}_{\Delta_f})$
6. Choose a small prime r such that $(\Delta_{\mathbb{K}}/r) = 1$
7. Set $\mathfrak{r}$ to a prime ideal of $\mathcal{O}_{\Delta_{\mathbb{K}}}$ lying above r
8. Pick $k \xleftarrow{\$} (\mathbb{Z}/f\mathbb{Z})^\star$ and set $\mathfrak{g} \leftarrow [\psi(\mathfrak{r}^2)] \cdot \mathfrak{f}^k$ in $C(\mathcal{O}_{\Delta_f})$
9. Set $B \leftarrow f \cdot \left\lceil \frac{\log(|\Delta_{\mathbb{K}}|) \cdot \sqrt{|\Delta_{\mathbb{K}}|}}{4\pi} \right\rceil$
10. Return $(B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F})$

sent to Solve as one of its inputs. Thus, the task of the Solve algorithm is to compute m from $\mathfrak{f}^m$. From Theorem 3.3.1 and Subsection 2.1.2, we know that $\mathfrak{f}^m$ has the form $(f^2, L(m)f)$. Solve takes $\mathfrak{f}^m$ and f as inputs and produces the exponent m as its output by computing $L(m)^{-1} \pmod f$. Algorithm 3.3.2 describes the pseudocode for the Solve algorithm.

Algorithm 3.3.2 Solve

Input: $\mathfrak{f}^m, f$

Output: m such that $\mathfrak{f}^m = (f^2, f)$

1. Parse $\mathfrak{m}$ as $(f^2, L(m)f)$
2. Return $L(m)^{-1} \pmod f$

Thus, summarising the instantiation, Algorithm 3.2.1 – Algorithm 3.2.5 present the linearly homomorphic encryption system first given in [12]. While we use the notation associated with the specific setting of class groups, this description applies to the generic setting of a DDH group with an easy DL subgroup of Definition 3.2.1. Here, plaintexts are integers modulo a prime conductor f, while ciphertexts are pairs of elements in $\mathcal{G}$. The size of the message space is completely determined by the size of the subgroup $\mathcal{F}$; in the class group

setting, the size of the subgroup $\mathcal{F}$ is precisely the conductor f of the non-maximal order $\mathcal{O}_{\Delta_f}$ [11, Lemma 1] and [17, Proposition 7.22 and Theorem 7.24]. Since the encryption algorithm computes ciphertexts c_1, c_2 in $C(\mathcal{O}_{\Delta_f})$, Encrypt performs two exponentiation steps and one multiplication step in $C(\mathcal{O}_{\Delta_f})$. We don't count the exponentiation $\mathfrak{f}^m = [(p^2, L(m)^{-1}p)]$ as it's simply the computation of the multiplicative inverse modulo $f = p$ as shown in Theorem 3.3.1. Thus, the most expensive computation in Encrypt are the two exponentiation steps, $\mathfrak{g}^r$ and $\mathfrak{h}^r$, in $C(\mathcal{O}_{\Delta_f})$. The most expensive computation in Decrypt is again the one exponentiation step, c_1^x, in $C(\mathcal{O}_{\Delta_f})$.

3.3.1 Security

In this section, we will present some results regarding the security of the generic LHE scheme presented by Castagnos and Laguillaumie and its specific instantiation in the class group setting. Note that in the scheme of a DDH group with an easy DL subgroup, the order n of the group $\mathcal{G}$, is not known [12, Definition 1]. If n was known, then one can compute a partial discrete logarithm $x \pmod s$ and compute $x \pmod n$ as p is known and totally break the scheme as shown in [12, Section 3.2]. The partial discrete logarithm (PDL) problem was introduced by Paillier in [35] and adapted to this specific class group setting by Castagnos and Laguillaumie.

Definition 3.3.1. Let $\mathcal{G}$ be a DDH group of order n with an easy DL subgroup $\mathcal{F}$ of order p and $\mathfrak{g}$ a generator of $\mathcal{G}$. Let $x \xleftarrow{\$} \mathbb{Z}/n\mathbb{Z}$. The Partial Discrete Logarithm (PDL) problem consists of computing $x \pmod p$, given $\mathfrak{g}$ and $\mathfrak{g}^x$ and access to the Solve algorithm.

Theorem 3.3.2. *[12, Lemma 1] Let $\mathcal{G}$ be a DDH group with an easy DL subgroup $\mathcal{F}$. The PDL problem reduces to the problem of computing the order n of G in polynomial time.*

Castagnos and Laguillaumie show that if an adversary $\mathcal{A}$ can compute the PDL problem in polynomial time then $\mathcal{A}$ can also solve the DDH assumption in $\mathcal{G}$ with a non negligible advantage.

Theorem 3.3.3. *[12, Lemma 2] Let $\mathcal{G}$ be a DDH group with an easy DL subgroup $\mathcal{F}$. The DDH problem in $\mathcal{G}$ reduces to the PDL problem in polynomial time.*

The following lift Diffie-Hellman (LDH) problem is a variant of the computational Diffie-Hellman problem, first introduced in [8] and adapted by Castagnos and Laguillaumie in their generic group setting. Castagnos and Laguillaumie show that the LDH and the PDL problems are equivalent in a DDH group with an easy DL subgroup $\mathcal{F}$.

Definition 3.3.2. [12, Definition 3] Let $\mathcal{G}$ be a DDH group of order n with an easy DL subgroup $\mathcal{F}$ and $\mathfrak{g}$ a generator of $\mathcal{G}$. Let $x, y \xleftarrow{\$} \mathbb{Z}/n\mathbb{Z}$ and let $\pi\colon \mathcal{G} \to \mathcal{G}/\mathcal{F}$ be the canonical surjection. The Lift Diffie-Hellman (LDH) problem consists of computing $\mathfrak{g}^{xy}$, given $(\mathfrak{g}, \mathfrak{g}^x, \mathfrak{g}^y, \pi(\mathfrak{g}^{xy}))$ and access to the Solve algorithm.

Theorem 3.3.4. *[12, Theorem 1] Let $\mathcal{G}$ be a DDH group with an easy DL subgroup $\mathcal{F}$. The LDH problem reduces to the PDL problem in polynomial time and vice versa.*

In Section 3.3, we have seen maps that exist between groups $\mathcal{G}$ and $\mathcal{G}/F$. Using these maps, Castagnos and Laguillaumie show the following reduction.

Theorem 3.3.5. *[12, Theorem 2] The DL problem in $\mathcal{G}/\mathcal{F}$ reduces to the DL problem in $\mathcal{G}$ in a DDH group with an easy DL subgroup.*

We now present the security assumptions that hold in the generic LHE scheme.

Theorem 3.3.6. *[12, Theoerem 3] The scheme described through Algorithm 3.2.1 - Algorithm 3.2.5 is one-way under chosen plaintext attack (OW-CPA) if and only if the LDH problem is hard.*

Theorem 3.3.7. *[12, Theorem 4] The scheme described through Algorithm 3.2.1 – Algorithm 3.2.5 is semantically secure under chosen plaintext attacks (IND-CPA) if and only if the DDH problem is hard in $\mathcal{G}$.*

From the description of the Gen algorithm *i.e.*, Algorithm 3.3.1, it is clear that one can compute the secret key x by solving the DL problem in the group $\mathcal{G}$. From the class group instantiation, this means that the group $\mathcal{G}$ is now a subgroup of $C(\mathcal{O}_{\Delta_p})$. Since the order of the group $\mathcal{G}$ is unknown and due to the size of the bound Bf on the order of $\mathcal{G}$, generic group algorithms to compute discrete logarithms in $\mathcal{G}$ are slower in practice. Also, Castagnos and Laguillaumie showed in [12, Section 3.2] that given oracles for solving the discrete logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ and computing the class number $h(\mathcal{O}_{\Delta_\mathbb{K}})$, one can solve the discrete logarithm in $C(\mathcal{O}_{\Delta_p})$ and compute the secret key x but no efficient algorithm currently exists that computes the class number $h(\Delta_\mathbb{K})$ or odd factors of it [12, Appendix B.3]. In [23], Hamdy and Möller discuss the selection criteria of the fundamental discriminant $\Delta_\mathbb{K}$ such that the discrete logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ is at least as hard as in finite fields. From these selection criteria described in Section 2.2, we see that the even factor of the class number is 2 from Step 1 in Algorithm 3.3.1. Since s denotes the size of the odd part of the class group, $s = h(\Delta_\mathbb{K})/2$. From Theorem 3.3.2 and Theorem 3.3.3, we have seen that if the order n of the group $\mathcal{G}$ is known, then one can easily solve the PDL in $\mathcal{G}$ and break the one-wayness of the scheme. Since $n = ps$ by construction of the Gen algorithm presented in Algorithm 3.3.1, it is obvious that s, the size of the odd part of the class group or equivalently, the odd factor of the class number $h(\Delta_\mathbb{K})$, should remain unknown and is the security parameter of the LHE scheme in the class group setting. To ensure that the group $\mathcal{G}$ is indeed of order $n = ps$ based on the description of Gen, it is important that p and s are relatively prime *i.e.*, $\gcd(p, s) = 1$.

In their construction, Castagnos and Laguillaumie assume p and s are relatively prime by ensuring p is of size at least 80 bits. This assumption is reasonable by the Cohen-Lenstra heuristics presented in Section 2.2. If p is at least 80 bits in length, then the probability of p dividing the class number and hence s is negligible. In the following chapter, we will show that this bound on p, *i.e.*, on the conductor, is not necessary. In Subsection 2.2.3, we discussed how one can compute discrete logarithms in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ and saw that one of these

methods was an index calculus method presented by Jacobson in [25]. We also saw that to provide b bits of security against the index-calculus method of computing the discrete logarithm in $C(\mathcal{O}_{\Delta_\mathbb{K}})$, one needs to select a fundamental discriminant $\Delta_\mathbb{K}$ of appropriate bit length. In [6, Table 4], Biasse *et al.* provide bit lengths of $\Delta_\mathbb{K}$ that are required to protect against the index calculus attacks in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ at various security levels. They also present the bit lengths of the RSA moduli that provide the same level of security at a given security level. We present some of their results, relevant to our thesis, in Table 3.1.

Also, the best known algorithms to compute the class number $h(\Delta_\mathbb{K})$ are known to have the same complexity as the corresponding discrete log algorithms in $C(\mathcal{O}_{\Delta_\mathbb{K}})$.

Table 3.1: Parameter sizes (in bits)

Security level	RSA Modulus	$\Delta_\mathbb{K}$
128	3072	1828
192	7680	3598
256	15360	5972

3.4 Short Cipher Variant

Castagnos and Laguillaumie proposed a variation of their scheme from Section 3.3 that uses ciphertexts that are smaller in size compared to the ones used in their basic cryptosystem. In Algorithm 3.2.1, we see that the generator $\mathfrak{g}$ of the group $\mathcal{G}$ is constructed as an element of $C(\mathcal{O}_{\Delta_f})$. In this new scheme, which we are going to call 'ShortCipher', Castagnos and Laguillaumie modify this step to construct $\mathfrak{g}$ in $C(\mathcal{O}_{\Delta_\mathbb{K}})$ as shown by Step 8 in Algorithm 3.4.1. The rest of the Gen subroutine for ShortCipher remains the same as Algorithm 3.3.1

Since $\mathfrak{g}$ is now in $C(\mathcal{O}_{\Delta_\mathbb{K}})$, the ciphertext c_1 in Encrypt for this variant is now in $C(\mathcal{O}_{\Delta_\mathbb{K}})$. The second ciphertext c_2 still needs to be constructed in $C(\mathcal{O}_{\Delta_f})$ as Theorem 3.3.1 holds in a non-maximal class group. To compute c_2, Castagnos and Laguillaumie now lift $\mathfrak{h}^r$ to $C(\mathcal{O}_{\Delta_f})$

Algorithm 3.4.1 Gen – ShortCipher

Input: $\lambda,\ \mu$ with $\lambda \geq \mu + 2$.

Output: $B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F}$

1. Pick random integers p and q such that p is a μ-bit prime, q is a $(2\lambda - \mu)$-bit prime, $pq \equiv 3 \pmod 4$ and $(p/q) = (q/p) = -1$
2. Set $\Delta_{\mathbb{K}} \leftarrow -pq$
3. Set $f \leftarrow p$
4. Set $\Delta_f \leftarrow f^2 \Delta_{\mathbb{K}}$
5. Set $\mathfrak{f} \leftarrow [(f^2, f)]$ in $C(\mathcal{O}_{\Delta_f})$
6. Choose a small prime r such that $(\Delta_{\mathbb{K}}/r) = 1$
7. Set $\mathfrak{r}$ to a prime ideal of $\mathcal{O}_{\Delta_{\mathbb{K}}}$ lying above r
8. Set $\mathfrak{g} \leftarrow [\mathfrak{r}^2]$ in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$
9. Set $B \leftarrow f \cdot \left\lceil \frac{\log(|\Delta_{\mathbb{K}}|) \cdot \sqrt{|\Delta_{\mathbb{K}}|}}{4\pi} \right\rceil$
10. Return $(B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F})$

by computing $\psi(\mathfrak{h}^r)$ where ψ is defined in Proposition 2.1.8. Thus, Encrypt now performs two exponentiation steps in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ and one lift whose cost is essentially one exponentiation in $C(\mathcal{O}_{\Delta_f})$ as shown in Proposition 2.1.8. The encryption pseudocode for this variant is shown in Algorithm 3.4.2.

Algorithm 3.4.2 Encrypt – ShortCipher

Input: pk, message m

Output: Ciphertext (c_1, c_2)

1. Pick $r \xleftarrow{\$} \{0, \ \ldots, \ Bf - 1\}$
2. Compute $c_1 \leftarrow \mathfrak{g}^r$
3. Compute $c_2 \leftarrow \mathfrak{f}^m \psi(\mathfrak{h}^r)$
4. Return (c_1, c_2)

To retrieve the message m, the decryption algorithm now has to recover the message m from the ciphertexts c_1 and c_2. Since c_1 is an element of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ and c_2 an element of $C(\mathcal{O}_{\Delta_f})$, the decryption algorithm first lifts c_1^x to $C(\mathcal{O}_{\Delta_f})$ by computing $\psi(c_1^x)$ and then computes $c_2/\psi(c_1^x)$ to receive $\mathfrak{f}^m$ which is reduced of form (f^2, f). Once it retrieves $\mathfrak{f}^m$, it

can now make a call to Solve using $\mathfrak{f}^m$ and recover the message m. Thus, the decryption algorithm in this variant now performs one exponentiation in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$, the computation of c_1^x, since c_1 is an element of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ and one exponentiation in $C(\mathcal{O}_{\Delta_f})$, the ψ map computation as shown in Proposition 2.1.8. The decryption pseudocode for this variant is described in Algorithm 3.4.3.

Algorithm 3.4.3 Decrypt – ShortCipher

Input: pk, sk, c_1, c_2

Output: Message m

1. Compute $\mathfrak{m} \leftarrow c_2/\psi(c_1^x)$
2. $m \leftarrow \mathsf{Solve}(B, f, \mathfrak{g}, \mathfrak{f}, \mathfrak{m})$
3. Return m

The semantic security of this variant is now based on a variant of the DDH problem. Let $\mathcal{G}$ be a subgroup of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ of order s and generated by $\mathfrak{g}$. Let $\mathfrak{g}^x, \mathfrak{g}^y$ be elements of $\mathcal{G}$ where $x, y \xleftarrow{\$} \mathbb{Z}/s\mathbb{Z}$. The DDH problem now consists of deciding whether $\psi(\mathfrak{g}^{xy}) = \psi(\mathfrak{g}^{xy})\mathfrak{f}^m$ where $\psi\colon C(\mathcal{O}_{\Delta_{\mathbb{K}}}) \to C(\mathcal{O}_{\Delta_f})$ is the map defined in Section 3.3, $\mathfrak{f}$ is a cyclic subgroup of $\mathcal{G}$ of order p and $m \xleftarrow{\$} \{1, \ldots, p-1\}$.

From this point forward, we will denote the schemes presented in Section 3.3 and Section 3.4 as 'CL schemes', the specific scheme presented in Section 3.3 as 'LongCipher' scheme and the scheme presented in Section 3.4 as 'ShortCipher'.

3.5 Handling Longer Message Lengths

Consider the cryptosystem presented in Section 3.3. The decryption algorithm works if and only if $\mathfrak{f}$ is reduced of form $[(f^2, f)]$. From Proposition 2.1.5, one can see that this imposes a bound on the values that f can take and thus, the possible sizes of the message space that one can use with the Castagnos and Laguillaumie cryptosystem is restricted. We will now try to find this restriction. From Subsection 2.1.2, it is clear that the ideal $\mathfrak{f} = [(f^2, f)]$ has

norm f^2. To guarantee reducedness, Proposition 2.1.5 gives the sufficient condition

$$\begin{aligned} f^2 &< \sqrt{|\Delta_f|}/2 \\ &= f\sqrt{|\Delta_{\mathbb{K}}|}/2 \text{ which holds if and only if} \\ \log_2 f &< \frac{1}{2}\log_2 |\Delta_{\mathbb{K}}| - 1 \end{aligned}$$

The bit length of f is thus determined by the following chain of equivalences:

$$\begin{aligned} \log_2 f &< \frac{1}{2}\log_2 |\Delta_{\mathbb{K}}| - 1 \\ \lfloor \log_2 f \rfloor &\leq \frac{1}{2}\lfloor \log_2 |\Delta_{\mathbb{K}}| \rfloor - 1 \\ bl(f) &\leq \frac{1}{2}\lfloor \log_2 |\Delta_{\mathbb{K}}| \rfloor \\ bl(f) &\leq \frac{1}{2}\lfloor \log_2 |\Delta_{\mathbb{K}}| \rfloor + \frac{1}{2} - \frac{1}{2} \\ bl(f) &\leq \frac{1}{2} bl(|\Delta_{\mathbb{K}}|) - \frac{1}{2} \end{aligned} \tag{3.1}$$

where $bl(x) = \lfloor \log_2(x) \rfloor + 1$ denotes the bit length of a positive integer x. We denote the quantity on the right-hand side of equation (3.1) by $bl(f)_{\max}$

Equation (3.1) shows that the largest size of the message space possible with the Castagnos and Laguillaumie scheme using the one step modular inverse decryption shown in Algorithm 3.3.2 is approximately half the bit length of $|\Delta_{\mathbb{K}}|$. Now consider a factorisation based linearly homomorphic scheme such as the Paillier cryptosystem (abbreviated Paillier) [35] or the Bresson *et al.* cryptosystem (abbreviated BCP) [8]. The size of the message space in these cryptosystems is the RSA modulus. Thus, if we are to compare the message spaces available among the Paillier, BCP and CL schemes using the information from Table 3.1, for example at the 128-bit security level, Paillier and BCP can use a messages of bit length 3072 bits where as CL can only use 912 bits from Equation (3.1). Ideally, we would like to be able to encrypt messages of bit lengths at least 1828 bits as Table 3.1 shows that a fundamental discriminant of size 1828 bits is enough to provide 128 bits of security. If one wished to

encrypt a message of bit length 3072 with the CL schemes, then using Equation (3.1), $|\Delta_{\mathbb{K}}|$ has to be at least 6146 bits, far larger than what is required at the same security level. As a result, the CL schemes discussed so far lose their advantage (a smaller security parameter) over factoring based schemes.

To overcome this disadvantage, Castagnos and Laguillaumie proposed a modification to their schemes that uses $\Delta_{\mathbb{K}} = -f$ with $f = p$ in Gen (Algorithm 3.3.1). Using $\Delta_{\mathbb{K}} = -f$ solves the problem of achieving message lengths equal to the bit length of $|\Delta_{\mathbb{K}}|$ but as one might correctly guess, this results in ideals of norm p^2 no longer being reduced in $C(\mathcal{O}_{\Delta_f})$ as they no longer satisfy Proposition 2.1.5. In order to still guarantee a polynomial time Solve algorithm, one solution they propose is to lift $\mathfrak{f} = [(f^2, f)]$ and its powers to the order $\mathcal{O}_{\Delta_{f^2}}$ of discriminant $\Delta_{f^2} = f^2\Delta_f$ where the lifted ideals are reduced since $f^2 < \sqrt{|\Delta_{f^2}|}/2$ if $|\Delta_K| > 4$. Castagnos and Laguillaumie show that $\mathfrak{f}_l$ belongs to the cyclic subgroup of $C(\mathcal{O}_{\Delta_{f^2}})$ generated by $[(f^2, f)]$ where $\mathfrak{f}_l = \psi(\mathfrak{f}) = [(f^2, f)]^z$ is the lift of $\mathfrak{f}$ under the lifting map $\psi\colon \mathfrak{f} \mapsto [(f^2, f)]^z$ that maps elements in $\mathcal{O}_{\Delta_f}$ to elements in $\mathcal{O}_{\Delta_{f^2}}$ [12, Section 4.1]. Thus, we can precompute the discrete logarithm z of $\mathfrak{f}_l$ with respect to $[(f^2, f)]$ in Gen using Solve but computing inside $C(\mathcal{O}_{\Delta_{f^2}})$. Note that this technique works for any $1 \leq q < 4p$ as well.

To recover a message using this technique, we first compute $c_2/c_1^x = \mathfrak{f}^m$ in Decrypt and then send $\mathfrak{f}^m$ to Solve-Lift as input. Solve-Lift lifts $\mathfrak{f}^m$ from $C(\mathcal{O}_{\Delta_f})$ to $C(\mathcal{O}_{\Delta_{f^2}})$ using the ψ map where $\psi\colon \mathfrak{f}^m \mapsto [(f^2, f)]^y$. Since $mz = y \pmod f$, one can then recover m as y/z where z is the discrete logarithm of $\psi(\mathfrak{f})$ with respect to the base $[(f^2, f)]$ and y is the discrete logarithm of $\psi(\mathfrak{f}^m)$ with respect to the basis $[(f^2, f)]$. Thus, this decryption technique, shown in Algorithm 3.5.1, adds one more exponentiation in $C(\mathcal{O}_{\Delta_{f^2}})$ during the decryption process.

It is important to note that Algorithm 3.5.1 can be used with both the LongCipher scheme as well as the ShortCipher scheme described in Section 3.4 to allow message spaces as big as the security parameter in the Castagnos and Laguillaumie cryptosystem, just like BCP and Paillier cryptosystems.

Algorithm 3.5.1 Solve – Lift

Input: $\mathfrak{f}^m, f, z$

Output: m such that $\mathfrak{f}^m = (f^2, L(m)f)$

1. Compute $\mathfrak{m} \leftarrow \psi(\mathfrak{f}^m)$
2. Parse $\mathsf{Red}(\mathfrak{m})$ as $(f^2, \tilde{y}f)$
3. Return $(\tilde{y}z)^{-1} \pmod{f}$

3.6 Conductor Choices

Castagnos and Laguillaumie used a prime conductor p in their schemes. However, towards the end of their paper, they mentioned that one can also use non prime conductors in their schemes for potential performance improvements and/or for larger message space sizes. Specifically, they suggested two forms of non prime conductors, the first being $f = p^t$ where p is a prime and t a positive integer and the second being $f = p_1 p_2 \cdots p_N$ where p_i are distinct primes and $N > 1$ is a positive integer. The suggested the use of a product of primes conductor in order to potentially improve the efficiency of the scheme using the Chinese Remainder Theorem and they suggested a prime power conductor to allow the size of the message space to be increased arbitrarily without increasing the security level.

In the following chapter, we will show how we used their proposed conductor forms to extend their scheme using conductors of form $f = (p_1 p_2 \cdots p_N)^t$ where $N, t \geq 1$.

3.7 Summary

In this chapter, we looked at the encryption schemes by Castagnos and Laguillaumie. The ShortCipher scheme computed one of the two ciphertexts in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ compared to LongCipher, which computed both the ciphertexts in $C(\mathcal{O}_{\Delta_f})$. Castagnos and Laguillaumie also proposed a modification to their scheme to use a conductor as big as the fundamental discriminant which increased the size of the message space that can be used with these schemes. In Table 3.2, we summarise the cost of these various schemes in terms of the number of

exponentiation steps they perform during encryption and decryption. The table provides the number of exponentiation steps carried out in various class groups during encryption (abbreviated E) and decryption (abbreviated D). We present this information for both the schemes and along with the lift modification presented in Section 3.5 that can be applied in each of these variants (the modified encryption and decryption abbreviated E-Lift and D-Lift respectively). Castagnos and Laguillaumie discussed open problems at the end of their paper that talked about choosing an alternate form of conductor f to possibly improve the efficiency of the schemes presented by them which we are going to discuss in the following chapter.

Table 3.2: Number of ideal exponentiation steps in each variant

Function	LongCipher			ShortCipher		
	$C(\mathcal{O}_{\Delta_{\mathbb{K}}})$	$C(\mathcal{O}_{\Delta_p})$	$C(\mathcal{O}_{\Delta_{p^2}})$	$C(\mathcal{O}_{\Delta_{\mathbb{K}}})$	$C(\mathcal{O}_{\Delta_p})$	$C(\mathcal{O}_{\Delta_{p^2}})$
E	0	2	0	2	1	0
E-Lift	0	2	0	2	1	0
D	0	1	0	1	1	0
D-Lift	0	1	1	1	1	1

Chapter 4

Using Conductor $f = (p_1 p_2 \cdots p_N)^t$

4.1 Introduction

The original probabilistic encryption scheme in [12] and the other variants presented in Chapter 3 use conductor $f = p$ where p is a prime. As mentioned in Section 3.6, Castagnos and Laguillaumie also suggested the use of a composite conductor f to potentially improve the efficiency of the variants and to allow the size of the message space to be increased arbitrarily without increasing the security level (governed by the size of the fundamental discriminant $\Delta_{\mathbb{K}}$). Specifically, they proposed $f = \prod_{i=1}^{N} p_i$ or $f = p^t$ where $N, t \in \mathbb{Z}_{\geq 1}$ and p_i, p are primes. In this section, we describe modified versions of the algorithms presented in Chapter 3 for the more general conductor $f = \prod_{i=1}^{N} p_i^t$ that includes the two proposed forms as the special cases $t = 1, N > 1$ and $t > 1, N = 1$ and the prime conductor p used by Castagnos and Laguillaumie as the case $N = t = 1$. We will see that the decryption technique differs depending if $t = 1$ or $t > 1$ and as such we will discuss them separately.

4.2 $N \geq 1$ and $t = 1$

Let $f = \prod_{i=1}^{N} p_i^t$ where $N, t \geq 1$ denote our generic conductor form. If $t = 1$, we have $f = \prod_{i=1}^{N} p_i$. We now need to ensure that the subgroup $\mathcal{F}$ is of size f as the size of the group

$\mathcal{F}$ is the size of the message space. Following the proof of Theorem 3.3.1 and [17, Theorem 7.24], we set the fundamental discriminant to be $\Delta_K = -p_1 p_2 \cdots p_N q$, $q \geq 1$, to ensure that the size of the group $\mathcal{F}$ is f, the size of our message space. From our discussion of the security of the CL scheme in Subsection 3.3.1, we see that $p_1, p_2, \cdots p_N, q$ should satisfy some Legendre symbol conditions discussed in Subsection 2.2.4 to ensure that the 2-Sylow subgroup of the class group $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ is as small as possible. Since the fundamental discriminant $\Delta_{\mathbb{K}}$ is now of form $\Delta_{\mathbb{K}} = -p_1 p_2 \cdots p_N q$, they satisfy the Legendre symbol conditions mentioned in Step 1 of Algorithm 3.3.1 because this ensures the 2-Sylow subgroup is of least possible size from Subsection 2.2.4.

Algorithm 4.2.1 Gen – using $f = (p_1 p_2 \cdots p_N)^t$

Input: $\lambda,\ \mu$

Output: $B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F}$

1. Pick random primes $p_1, p_2, \cdots, p_N, q$ such that $p_1 p_2 \cdots p_N$ is a μ-bit integer, q is a $(2\lambda - \mu)$-bit prime, $p_1 p_2 \cdots p_N q \equiv 3 \pmod 4$ and $(p_i/p_j) = (p_j/p_i) = 1$ and $(p_i/q) = (q/p_i) = -1$ for $1 \leq i, j \leq N$
2. Set $\Delta_{\mathbb{K}} \leftarrow -p_1 p_2 \cdots p_N q$
3. Set $f \leftarrow (p_1 p_2 \cdots p_N)$
4. Set $\Delta_f \leftarrow f^2 \Delta_{\mathbb{K}}$
5. Set $\mathfrak{f} \leftarrow [(f^2, f)]$ in $C(\mathcal{O}_{\Delta_f})$
6. Choose a small prime r such that $\gcd(r, f) = 1$ and $(\Delta_{\mathbb{K}}/r) = 1$
7. Set $\mathfrak{r}$ a prime ideal lying above r
8. Pick $k \xleftarrow{\$} (\mathbb{Z}/f\mathbb{Z})^\star$
9. Set $\mathfrak{g} \leftarrow [\psi(\mathfrak{r}^2)] \cdot \mathfrak{f}^k$ in $C(\mathcal{O}_{\Delta_f})$
10. Set $B \leftarrow |\mathcal{M}| \cdot \left\lceil \frac{\log(|\Delta_{\mathbb{K}}|) \cdot \sqrt{|\Delta_{\mathbb{K}}|}}{4\pi} \right\rceil$
11. Return $(B, f, \mathfrak{g}, \mathfrak{f}, \mathcal{G}, \mathcal{F})$

We now look at the decryption techniques that one can use with this conductor form. If we assume $q > 4f$ as before, then the reduced representative of the ideal class $\mathfrak{f} \in C(\mathcal{O}_{\Delta_f})$ is $[(f^2, f)]$ and $\mathfrak{f}$ generates a cyclic group of order f in $C(\mathcal{O}_{\Delta_f})$. This is shown in Theorem 3.3.1 for $f = p$ and this result carries over to $f = (p_1 p_2 \cdots p_N)^t$ via Chinese remaindering. We now

present ways to encrypt and decrypt a message m using this form of conductor. Note that one can now recover m by performing the inverse modulo the individual prime factors of f and then use the Chinese Remainder Theorem (CRT) to solve the system of congruences $m \equiv m_i \pmod{p_i}$ where $1 \leq i \leq N$. This can be done in three ways: The first CRT modification is straightforward: The two decryption techniques presented by Castagnos and Laguillaumie, decryption in $C(\mathcal{O}_{\Delta_f})$ and $C(\mathcal{O}_{\Delta_{f^2}})$, compute $L(m)^{-1} \pmod{f}$ where $\mathfrak{f}^m = [(f^2, L(m)f)]$ and $mL(m) \equiv 1 \pmod{f}$. The first CRT method we describe is to compute $L(m)^{-1} \pmod{p_i}$ for all i instead of $L(m)^{-1} \pmod{f}$ and using the Chinese Remainder Theorem to recover m. This modification involves no exponentiation just like Algorithm 3.3.2 and instead computes the inverse modulo smaller primes as shown in Algorithm 4.2.2.

Algorithm 4.2.2 Decrypt – CRT 1

Input: $f, \mathfrak{f}, \mathfrak{m}$

Output: m such that $\mathfrak{m} = \mathfrak{f}^m$

1. Parse $\texttt{Red}(\mathfrak{m})$ as $(f^2, \tilde{x}f)$
2. Compute $m_i \leftarrow \tilde{x}^{-1} \pmod{p_i}$ where $1 \leq i \leq N$
3. Solve $m \equiv m_i \pmod{p_i}$ where $1 \leq i \leq N$
4. Return m

Before we present the next two methods, note that when $f = \prod_{i=1}^{N} p_i$, $\mathcal{F}$ contains order p_i subgroups for each i that are generated by the elements $\mathfrak{f}_i = \mathfrak{f}^{(f/p_i)}$ and are represented by the ideals (p_i^2, p_i). We can precompute these $\mathfrak{f}_i$'s during the key generation process and use them instead of $\mathfrak{f}$ in the encryption algorithm.

In the second CRT method, we compute $m_i \equiv m \pmod{p_i}$ and compute the second ciphertext c_2 as $\mathfrak{f}_1^{m_1}\mathfrak{f}_2^{m_2}\cdots\mathfrak{f}_N^{m_N}\mathfrak{h}^r$. This is a valid encryption as c_2 is still an element of $C(\mathcal{O}_{\Delta_f})$. With this encryption, we are only performing 2 exponentiations in $C(\mathcal{O}_{\Delta_f})$ just like Algorithm 3.3.2 as the cost of computing $\mathfrak{f}_i^{m_i}$ is essentially one modular inversion. We present this encryption algorithm in Algorithm 4.2.3.

To retrieve the message m, the decryption algorithm now computes $\mathfrak{f}^m = c_2/c_1^x$ and parses the result as $(f^2, \tilde{x}f)$, just like in Algorithm 3.2.3. However, the representation of $\mathfrak{f}^m$ is no

Algorithm 4.2.3 Encrypt – CRT 2

Input: pk, m

Output: Ciphertext (c_1, c_2)

1. Pick $r \xleftarrow{\$} \{0, \cdots, Bf - 1\}$
2. Compute $m_i \leftarrow m \pmod{p_i}$
3. Compute $c_1 \leftarrow \mathfrak{g}^r$
4. Compute $c_2 \leftarrow \mathfrak{f}_1^{m_1}\mathfrak{f}_2^{m_2}\cdots\mathfrak{f}_N^{m_N}\mathfrak{h}^r$
5. Return c_1, c_2

longer $(f^2, L(m)f)$ with $mL(m) \equiv 1 \pmod{f}$ as $\mathcal{F}$ is now a product of N cyclic groups. Instead, we compute the m_i we computed during encryption as $(\tilde{x}f/p_i)^{-1} \pmod{p_i}$ and then use the Chinese Remainder Theorem to get m as we show in Algorithm 4.2.4.

Algorithm 4.2.4 Decrypt – CRT 2

Input: $f, \mathfrak{f}, \mathfrak{m}$

Output: m such that $\mathfrak{m} = \mathfrak{f}^m$

1. Parse $\texttt{Red}(\mathfrak{m})$ as $(f^2, \tilde{x}f)$
2. $m_i \leftarrow (\tilde{x}f/p_i)^{-1} \pmod{p_i}$
3. Solve $m \equiv m_i \pmod{p_i}$
4. Return m

The third CRT method uses the following observation: Since $\mathcal{F}$ contains subgroups of order p_i for $1 \leq i \leq N$, one could also encrypt $m_i \pmod{p_i}$ creating $N+1$ ciphertexts. This means that encryption now computes $\mathfrak{f}_i^{m_i}$ as $(p_i^2, L(m_i)p_i)$ where $m_iL(m_i) \equiv 1 \pmod{p_i}$, making this variant a viable option. This algorithm is shown in Algorithm 4.2.5.

To decrypt the ciphertexts, one simply computes $m_i \pmod{p_i}$ and solves the system of congruences $m = m_i \pmod{p_i}$ via the Chinese Remainder Theorem. We present the pseudo-code of this decryption in Algorithm 4.2.6. Note that the three CRT methods also work with the lift decryption method presented in Section 3.5.

Algorithm 4.2.5 Encrypt – CRT 3

Input: pk, m

Output: Ciphertext (c_1, c_2)

1. Pick $r \xleftarrow{\$} \{0, \cdots, Bf - 1\}$
2. Compute $m_i \leftarrow m \pmod{p_i}$
3. Compute $c_1 \leftarrow \mathfrak{g}^r$
4. Compute $\hat{c}_i \leftarrow \mathfrak{f}_i^{m_i} \mathfrak{h}^r$
5. Return $c_1, \hat{c}_i$

Algorithm 4.2.6 Decrypt – CRT 3

Input: $pk, sk, c_1, \hat{c}_1, \cdots, \hat{c}_n$

Output: Message m

1. Compute $\mathfrak{m}_i \leftarrow \hat{c}_i / c_1^x$
2. Compute $m_i \leftarrow \mathsf{Solve}(p_i, \mathfrak{f}_i, \mathfrak{m}_i)$
3. Solve $m \equiv m_i \pmod{p_i}$
4. Return m

4.3 $N \geq 1$ and $t > 1$

We have seen in Section 3.5 that in order to accommodate message space sizes bigger than $bl(f)_{\max}$, Castagnos and Laguillaumie perform decryption by lifting $\mathfrak{f}^m$ from $C(\mathcal{O}_{\Delta_f})$ to a smaller order $C(\mathcal{O}_{\Delta_{f^2}})$ with conductor f^2 to recover m using the one step modular inversion decryption method. They do this as (f^2, f) is no longer reduced in $C(\mathcal{O}_{\Delta_f})$ when $\Delta_{\mathbb{K}} = -p_1 \cdots p_N$ and $f = \prod_{i=1}^{N} p_i^t$. We now present a way where we perform the decryption process in $C(\mathcal{O}_{\Delta_f})$ itself without lifting to $C(\mathcal{O}_{\Delta_f})$. To do this, note that if $\mathcal{F} \subset C(\mathcal{O}_{\Delta_f})$ has order $f = \prod_{i=1}^{N} p_i^t$, it contains subgroups whose orders are p_i^t where each group of order p_i^t contains a cyclic subgroup of order p_i from Theorem 3.3.1. One can then compute the discrete logarithm m_i in subgroups of order p_i^t and then combine the m_i using the Chinese Remainder Theorem. To compute the discrete logarithm in subgroups of order p_i^t, we use the well known Pohlig-Hellman algorithm to iteratively compute the p-adic digits of the logarithm [37].

Given $\mathfrak{f} \in \mathcal{F}$ of order f and $\mathfrak{m} = \mathfrak{f}^m$ where f has prime factors $p_1, \cdots, p_N$, the Pohlig-Hellman subroutine first computes $\mathfrak{f}_i = \mathfrak{f}^{f/p_i^t}$ where $\mathfrak{f}_i$ has order p_i^t where $1 \leq i \leq N$. Next, it computes $\mathfrak{m}_i = \mathfrak{m}^{f/p_i^t}$, thus lifting $\mathfrak{f}$ and $\mathfrak{m}$ to subgroups of order p_i^t. It next iteratively computes the digits of the logarithm in subgroup p_i^t and base $\mathfrak{f}_i^{p_i^{t-1}}$ where the base has order p_i. Lastly, it combines all the discrete logarithms in each of p_i^t using CRT to get the message m. Thus, the decryption algorithm now performs $1 + N \cdot (3 + 2t)$ exponentiations in the non-maximal class group $C(\mathcal{O}_{\Delta_f})$ during the decryption process.

To use the Pohlig-Hellman subroutine in the decryption algorithm, the key generation and encryption algorithms remain the same as Algorithm 3.2.1, Algorithm 3.2.2 respectively, whereas the decryption algorithm now uses the Pohlig-Hellman subroutine as shown in Algorithm 4.3.1.

Algorithm 4.3.1 Decrypt – using Pohlig-Hellman

Input: $pk, sk, (c_1, c_2)$

Output: Message m

1. Compute $\mathfrak{m} = c_2/c_1^x$
2. **for** $i = 1$ to N **do**
3. Compute $\mathfrak{f}_i \leftarrow \mathfrak{f}^{f/p_i^t}$
4. Compute $\mathfrak{m}_i \leftarrow \mathfrak{m}^{f/p_i^t}$
5. Set $x_0 \leftarrow 0$
6. Compute $\gamma \leftarrow \mathfrak{f}_i^{p_i^{t-1}}$
7. **for** $k \leftarrow 0$ to $t-1$ **do**
8. Compute $\mathfrak{m}'_k \leftarrow \left(\mathfrak{m}_i \mathfrak{f}_i^{-x_k}\right)^{p_i^{t-1-k}}$
9. Compute $d_k \leftarrow \mathsf{Solve}(p_i, \gamma, \mathfrak{m}'_k)$
10. Set $x_{k+1} \leftarrow x_k + p^k d_k$
11. **end for**
12. Set $m_i \leftarrow x_t$
13. **end for**
14. Solve $m \equiv m_i \pmod{p_i^t} \quad \forall i \in \{1, \cdots, N\}$ using CRT
15. Return m

Note that we can precompute the $\mathfrak{f}_i \leftarrow \mathfrak{f}^{f/p_i^t}$ during the key generation process and solve discrete logarithms in subgroups of order p_i^t and then combine $m_i \pmod{p_i^t}$ using the Chinese Remainder Theorem. This is similar to the CRT 3 decryption mentioned in Section 4.2 and we present this variant below.

Algorithm 4.3.2 Encrypt – CRT 3

Input: pk, m

Output: Ciphertext (c_1, c_2)

1. Pick $r \xleftarrow{\$} \{0, \cdots, Bf - 1\}$
2. Compute $m_i \leftarrow m \pmod{p_i^t}$
3. Compute $c_1 \leftarrow \mathfrak{g}^r$
4. Compute $\hat{c}_i \leftarrow \mathfrak{f}_i^{m_i}\mathfrak{h}^r$
5. Return $c_1, \hat{c}_i$

Algorithm 4.3.3 Decrypt – CRT 3

Input: $pk, sk, (c_1, c_2)$

Output: Message m

1. Compute $\mathfrak{m}_i = \hat{c}_i / c_1^x$
2. Set $x_0 \leftarrow 0$
3. **for** $i = 1$ to N **do**
4. **for** $k \leftarrow 0$ to $t - 1$ **do**
5. Compute $\mathfrak{m}'_k \leftarrow \left(\mathfrak{m}_i \mathfrak{f}_i^{-x_k}\right)^{p_i{}^{t-1-k}}$
6. Parse $\mathfrak{m}'_k$ as $(p_i^2, \tilde{x} p_i)$
7. Set $d_k \leftarrow \tilde{x}^{-1} \pmod{p_i}$
8. Set $x_{k+1} \leftarrow x_k + p^k d_k$
9. **end for**
10. Set $m_i \leftarrow x_t$
11. **end for**
12. Solve $m \equiv m_i \pmod{p_i^t} \quad \forall i \in \{1, \cdots, N\}$ using CRT
13. Return m

We see that the encryption algorithm presented in Algorithm 4.3.2 performs 2 + N exponentiations where as the decryption algorithm presented in Algorithm 4.3.3 performs

Table 4.1: Parameter sizes (in bits)

			Δ_f		
Security level	RSA Modulus	$\Delta_{\mathbb{K}}$	$bl(f) = 16$	80	256
128	3072	1828	1860	1988	2340
192	7680	3598	3630	3758	4110
256	15360	5972	6004	6132	6484

$1 + N \cdot (0 + 2t)$ exponentiations in the non-maximal class group $C(\mathcal{O}_{\Delta_f})$.

4.4 Parameter Choices

As described in [12], the main concern with selecting parameters is that it should be computationally infeasible to compute $h(\Delta_{\mathbb{K}})$, the class number of the maximal order $\mathcal{O}_{\Delta_{\mathbb{K}}}$. This is because, from Subsection 3.3.1, knowledge of the class number solves the DDH assumption. Biasse *et al.* in [6] gave estimates of discriminant sizes to provide various levels of security using the best known index calculus algorithms of subexponential complexity. In Table 4.1, we give these sizes for the 128-, 192-, and 256-bit security levels, along with the corresponding RSA modulus sizes required for Paillier [35] and Bresson *et al.* [8]. Note that generic group algorithms do not play a role here, as their complexity is worse than the index calculus algorithms.

We also list in Table 4.1 the sizes of the non-fundamental discriminants Δ_f required to provide the given security level for various message sizes. As mentioned earlier, the basic variants of the Paillier and Bresson *et al.* cryptosystems can encrypt messages of up to the same size as the RSA modulus used, which is determined by the desired security level. The variants of the Castagnos and Laguillaumie cryptosystem have their security level fixed primarily by the size of the fundamental discriminant $\Delta_{\mathbb{K}}$, and can work with different message sizes by varying the conductor. In Chapter 5, we will see that smaller messages provide better runtimes than Paillier and BCP schemes at increasing security levels, as even the non-maximal discriminants are quite small compared to the RSA moduli required by

Paillier and Bresson *et al.* at the same security levels.

The two other considerations for security parameter choices are the sizes of primes dividing the conductor f and the upper bound for selecting random exponents in the protocol. We now discuss these two considerations.

4.4.1 Restrictions on Prime Factors of f

Castagnos and Laguillaumie insisted on using a conductor (a prime p) of size at least 80 bits to ensure $\gcd(p, h(\Delta_{\mathbb{K}})) = 1$ with high probability, implying that the odd part of the class number is completely unknown. This is important as the odd part of $h(\Delta_{\mathbb{K}})$ is the security parameter and knowing the odd part, s, leads to a total break of the scheme as shown in Subsection 3.3.1. If a divisor of s is known, then computing s itself may be easier. Extrapolating this idea to our extension in which the conductor is a product of prime powers would then imply that the prime divisors of the conductor be at least 80 bits. This restriction is an impediment to the performance of both the original and extended versions of the cryptosystem when one considers how a known factor of $h(\Delta_{\mathbb{K}})$ could be exploited in practice.

The best known algorithms to compute the class number are sub-exponential index-calculus algorithms and generic group algorithms from Section 2.2. There is no known way to speed the index calculus algorithms given a divisor of the class number, as the complexity depends on the discriminant as opposed to the class number. Thus, the discriminant sizes recommended by Biasse *et al.* in [6] offer enough protection against index calculus algorithms even if a divisor of the class number is known. When considering generic algorithms, on the other hand, a known divisor of the class number does improve the running time, as one can target the unknown part directly. We consider the worst case that the entire conductor f divides the odd part of the class number (note that f itself is odd). Let 2^{k_1} be the even and $s = f \cdot s'$ be the odd factors of $h(\Delta_{\mathbb{K}})$ where $k_1 \in \mathbb{Z}_{\geq 0}$ and s' is the unknown part of the odd

part s. Since $h(\Delta_{\mathbb{K}}) < \frac{1}{\pi}\log(|\Delta_{\mathbb{K}}|)\sqrt{\Delta_{\mathbb{K}}}$ from Equation (2.3), we have,

$$s' < \frac{1}{2^{k_1} \cdot f} \cdot \frac{1}{\pi}\sqrt{|\Delta_{\mathbb{K}}|}\log|\Delta_{\mathbb{K}}|. \tag{4.1}$$

Generic group algorithms can be used to compute s' in time $O(\sqrt{s'})$. Ignoring constants and lower-order terms, in order to provide b bits of security, we require that $\sqrt{s'} > 2^b$. Combining this with equation (4.1) yields the following upper bound on $\log_2 f$:

$$\log_2 f \leq \log_2 \log|\Delta_{\mathbb{K}}| + \frac{1}{2}\log_2|\Delta_{\mathbb{K}}| - k_1 - 2 - 2b. \tag{4.2}$$

For example, following the recommendations in [6], Castagnos and Laguillaumie chose a discriminant of size 1828 bits at the 128-bit security level to prevent index calculus attacks in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. Substituting these values in Equation (4.2) results in

$$\log_2 f \leq 667 - k_1,$$

meaning that we can tolerate known divisors of the conductor of size over 600 bits before the generic attacks would work in fewer than 2^b operations. Note that using conductors with prime divisors larger than this bound is also highly unlikely to be an issue, because, from Section 2.2 item 2, the probability that primes of this size divide the class number is negligible — indeed, in [12], using primes larger than 2^{80} was deemed to be sufficient. Thus, we conclude that based on the current state of knowledge of possible attacks on the cryptosystem, prime divisors of any size in the conductor are unlikely to result in any loss of security. This is because the discriminant sizes required to avoid index calculus attacks result in class numbers that are sufficiently large to prevent the generic algorithms, which could exploit a known factor of the class number, from working in fewer than 2^b operations.

4.4.2 Selection of Random Exponents

The bound B on exponents in Algorithm 3.3.1, taken directly from [12], is designed to ensure that the resulting group elements are selected from the entire class group uniformly at random, a necessary condition for the security proofs to hold. In [12], the formula for B has a factor of 2^{80} in order to ensure a statistical distance of 2^{-80} from the uniform distribution; in our exposition above, we instead use the size of the message space f, thus allowing the resulting statistical distance $1/f$ to vary with the size of the message space. In the following chapter, in which we benchmark the practical performance of their version as well as our extensions, we will consider this version of the cryptosystem.

However, it is also known that one can obtain similar security proofs using much shorter exponents if one is willing to use slightly non-standard versions of the intractability assumptions, a critical performance optimization. There is no known way to take advantage of knowledge that discrete logarithms are small in the index calculus algorithms, so these have no bearing on the exponent bounds. The only concern is with generic algorithms of square root complexity, which imply that all exponents should be chosen with at least $2b$ bits for a b-bit security level. Koshiba and Kurosawa [30] proved that security proofs relying on Diffie-Hellman problems also hold assuming that such short exponent versions of the discrete logarithm problem are intractable. Thus, it is at least plausible that the security proofs of [12] also hold under similarly modified intractability assumptions. We will also consider short exponent versions of our cryptosystems, as is typically done in practice, in the benchmarks presented in the next section, using exponents of $2b$ bits.

4.5 Summary

In this chapter, we replaced the prime conductor p in LongCipher and ShortCipher from Section 3.7 with $f = (p_1 p_2 \cdots p_N)^t$. We presented revised algorithms where necessary to accommodate this new conductor form. Following Castagnos and Laguillaumie's suggestions,

we also presented new algorithms using Chinese Remaindering to improve the performance of the CL schemes. Additionally, we eliminated the unnecessary restriction on prime factors of f that was imposed by Castagnos and Laguillaumiein [12]. We used previous work by Koshiba and Kurosawa [30] to use random exponent sizes just large enough to beat practical discrete logarithm attacks in class groups. In Table 4.2, we present a summary of the number of exponentiations carried out during encryption (E) and decryption (D) in various class groups. The three CRT modifications presented in Section 4.2 are abbreviated CRT 1, CRT 2 and CRT 3 respectively.

Table 4.2: Number of ideal exponentiations in each variant

t	Function	LongCipher		ShortCipher	
		$C(\mathcal{O}_{\Delta_{\mathbb{K}}})$	$C(\mathcal{O}_{\Delta_f})$	$C(\mathcal{O}_{\Delta_{\mathbb{K}}})$	$C(\mathcal{O}_{\Delta_f})$
1	E	0	2	2	1
	E CRT 2	0	2	2	1
	E CRT 3	0	2	2	1
	D	0	1	1	1
	D CRT 1	0	1	1	1
	D CRT 2	0	1	1	1
	D CRT 3	0	1	1	1
> 1	E	0	3	2	2
	E CRT 3	0	$2 + N$	2	$1 + N$
	D	0	$1 + N(3 + 2t)$	1	$1 + N(3 + 2t)$
	D CRT 3	0	$1 + N(0 + 2t)$	1	$1 + N(0 + 2t)$

Chapter 5

Results

In Chapter 3, we presented two encryption schemes by Castagnos and Laguillaumie. We saw that these schemes used a prime conductor p and were based in class groups of imaginary quadratic fields. We presented the second scheme, ShortCipher, in Section 3.4, which was a variant of the first scheme, LongCipher. ShortCipher used ciphertexts which were smaller in size compared to the ones used in LongCipher. In Chapter 4, we extended these schemes, based on the recommendations made by Castagnos and Laguillaumie, by using a composite conductor of form $(p_1 p_2 \cdots p_N)^t$ where p_i are primes and N and t are integers with $N, t \geq 1$. As a result, we were able to use the Chinese Remainder Theorem to improve the runtime performance of the encryption schemes when $t > 1$.

Castagnos and Laguillaumie implemented their schemes in Sage [41] using PARI/GP [36] for ideal arithmetic. They compared the results of their implementation with Sage implementations of the Paillier and the BCP schemes [12, Section 5]. Their results show that their own schemes were competitive in performance with the Paillier and BCP schemes but pointed out the presence of a bias against their own schemes. This is because Castagnos and Laguillaumie carried out ideal exponentiation using PARI/GP which was far less optimised than integer exponentiation modulo the RSA modulus n that is central to the Paillier and BCP schemes. As a result, Castagnos and Laguillaumie emphasised the need for a more

optimised implementation for perhaps better results in favour of their schemes.

In this chapter, we present the details of our optimised implementation of the schemes by Castagnos and Laguillaumie in C++, using a state-of-the-art implementation of ideal arithmetic due to Sayles [42] along with the GMP multiprecision integer library [21]. In order to ensure consistent and fair benchmarking, we also implemented the Paillier and BCP schemes using the same platform. We did this for our extensions of the CL schemes as well and present comparative results based on our implementation of the three encryption schemes.

5.1 Description of Experiments

5.1.1 Objectives

The objective of our experiments was twofold. First, we wanted to find the fastest encryption and decryption procedures among the CL schemes presented in Section 3.3 and Section 3.4 as well as our variations presented in Chapter 4. This benchmarking was performed for different combinations of three parameters, the bit length of the message to be encrypted, the security parameter, and the bit length of the random exponents. Our second goal was to compare the performance of the fastest versions of the CL schemes with the Paillier and BCP schemes for different levels of security and different message lengths at each level. We next discuss the methodology of our experiments performed to satisfy these objectives.

5.1.2 Methodology

For our benchmarking experiments, we implemented the CL, Paillier and BCP schemes using C++. We considered three security levels - 128, 192 and 256 bits, taking NIST recommendations [4] into consideration. For each of these security levels, we considered four conductor bit lengths (equivalently, message bit lengths) - 16, 80, 256 and 3072/7680/15360. Since results by Castagnos and Laguillaumie [12, Table 1] show their schemes to be faster than the

BCP and Paillier schemes for small message sizes, we chose message lengths of 16, 80 and 256 bits. Also, Castagnos and Laguillaumie promoted 80-bit messages for practical applications. Since our goal was to compare the CL schemes with the BCP and Paillier schemes, where the latter two allow message sizes up to the size of the RSA modulus used, our fourth choice of the conductor bit length stems from this objective. Table 3.1 and [4] showed that the sizes of the RSA moduli to provide 128, 192 and 256 bits of security are 3072, 7680 and 15360 bits respectively.

We now discuss the parameter generation process. We used conductors of the form $f = (p_1 \cdots p_N)^t$ where $N, t \geq 1$. For example, consider the 128-bit security level. From Table 3.1 we see that a 1828-bit discriminant $\Delta_{\mathbb{K}}$ is required at this security level. From Equation (3.1), we see that for conductors up to 912 bits in length, one can use the one step modular inversion method presented in Algorithm 3.3.2. Recall that $bl(x)$ denotes the bit length of x and $bl(f)_{\max}$ is the upper bound on the bit length of the conductor from Equation (3.1). The largest bit length of f that one can use with Algorithm 3.3.2 at a given security level. For conductors of bit lengths 16, 80 and 256, $\Delta_{\mathbb{K}} = -p_1 p_2 \cdots p_N q$ and $q > 1$ where $f = (p_1 p_2 \cdots p_N)^t$. As these message lengths are smaller than the threshold $bl(f)_{\max}$, we suspect that the one step decryption will be faster than any of the other algorithms we presented in Chapter 4. Preliminary experimental results seem to support this claim. Nevertheless, we timed all the algorithms applicable to this case for (N, t) pairs where N and t take values from the set $\{1, 2, 3, 4, 5\}$. From our preliminary observations, these (N, t) pairs should be more than sufficient to find the (N, t) pair that gives the best performance result. For message lengths larger than $bl(f)_{\max}$, it is clear that $q = 1$ and $t \geq \lceil (bl(f)/bl(|\Delta_{\mathbb{K}}|)) \rceil$ when $\Delta_{\mathbb{K}} = -p_1 p_2 \cdots p_N q$ and $f = (p_1 p_2 \cdots p_N)^t$. In this case, we varied only the values of N and fixed $t = \lceil (bl(f)/bl(|\Delta_{\mathbb{K}}|)) \rceil$.

In Table 5.1, we present a summary of parameter choices for the various cryptosystems at each security level. Column 1 describes the security level in bits followed by the name of the cryptosystem in Column 2. Columns 3–4 describe the bit lengths of the fundamental

discriminant $\Delta_{\mathbb{K}}$ and the RSA modulus n used in the corresponding cryptosystem, respectively. These values are taken from Table 3.1. Column 5 describes the message lengths in bits that were used in our experiments. Columns 6–7 describe the values of N and t for conductor $f = (p_1 p_2 \cdots p_N)^t$.

Table 5.1: Summary of Parameter Choices

Security	Cryptosystem	$bl(\Delta_{\mathbb{K}})$	$bl(n)$	$bl(f)$	N	t
128	CL	1828		$16, 80, 256$	$1, 2, \ldots, 5$	$1, 2, \ldots, 5$
				3072	$1, 2, \ldots, 24$	2
	BCP		3072	$16, 80, 256, 3072$		
	Paillier		3072	$16, 80, 256, 3072$		
192	CL	3598		$16, 80, 256$	$1, 2, \ldots, 5$	$1, 2, \ldots, 5$
				7680	$1, 2, \ldots, 24$	3
	BCP		7680	$16, 80, 256, 7680$		
	Paillier		7680	$16, 80, 256, 7680$		
256	CL	5972		$16, 80, 256$	$1, 2, \ldots, 5$	$1, 2, \ldots, 5$
				15360	$1, 2, \ldots, 24$	3
	BCP		15360	$16, 80, 256, 15360$		
	Paillier		15360	$16, 80, 256, 15360$		

To compare the performance of the CL schemes with the Paillier and BCP schemes, we also implemented the Paillier and BCP schemes in order to ensure a uniform testing platform. For the BCP scheme, we implemented the scheme presented in [8, Section 3] with the first decryption procedure and [35, LongCipher] for Paillier along with its CRT variant. We present the runtimes of the BCP and Paillier schemes in Appendix A, Appendix B, respectively and those of the CL schemes in Appendix C and Appendix D.

Our experiments were carried out on a standard desktop running on Intel Core i7-8700, at 3.20 GHz with 15.5 GiB RAM, running Fedora 29. Our programs were written in C and C++ (using `gcc` version 8.3.1) with GMP (version 6.1.2) [21] and NTL (version 11.3.2) [46] support for arbitrary precision arithmetic. We used Sayles's binary quadratic forms library `libqform` for ideal arithmetic and his `liboptarith` library for some optimised GMP integer type based arithmetic [43]. For ideal exponentiation, we tested several generic single

exponentiation methods such as standard binary, binary from left to right, non adjacent form (NAF) and its windowing variant wNAF [39] with values of w ranging from 3 to 10. We also tested double exponentiation methods such as joint sparse form [47], hybrid binary ternary joint sparse form [1] and interleaving wNAF methods [32] [33]. We observed that a window size of 8 gave the best single exponentiation results and the interleaving method with the same window size 8 gave the best double exponentiation results. However, the exponentiation computations of $\mathfrak{f} = [(f^2, f)]$ and $\mathfrak{f}_i = [(p_i^2, p_i)]$ were performed by simply computing the inverse of the exponent modulo the order of $\mathfrak{f}$ and setting $\mathfrak{f}^x$ as $[(f^2, x^{-1}f)]$ when $t = 1$. In Table 5.2, we present a summary of exponentiation algorithms used in our experiments.

Table 5.2: Summary of Exponentiation Algorithms used in CL schemes

t	Exponentiation(s)	Algorithm used
1	$\mathfrak{g}^x$	8NAF
	$\mathfrak{f}^x$ and $\mathfrak{f}_i^{x_i}$	Modular inversion
> 1	$\mathfrak{g}^x$ and $\mathfrak{f}^y$	8NAF
	$\mathfrak{f}^x \cdot \mathfrak{g}^y$ and $\mathfrak{f}_i^{x_i} \cdot \mathfrak{g}^y$	Interleaving 8NAF

5.2 Analysis of CL Schemes

In this section, we present the results of all our experiments. Appendix C contains the results of the implementation of the LongCipher scheme whereas Appendix D contains the results of the implementation of the ShortCipher scheme. Based on the results of these experiments, we now draw conclusions on the effect of different parameters of the performance of the schemes. We first discuss this for message lengths smaller than $bl(f)_{\max}$ and then for message lengths the size of the RSA modulus in the Paillier and Bresson *et al.* schemes. We use the same notation to describe our results as in Appendix C, page 94 and Appendix D, page 111.

5.2.1 Case $bl(f) < bl(f)_{\max}$

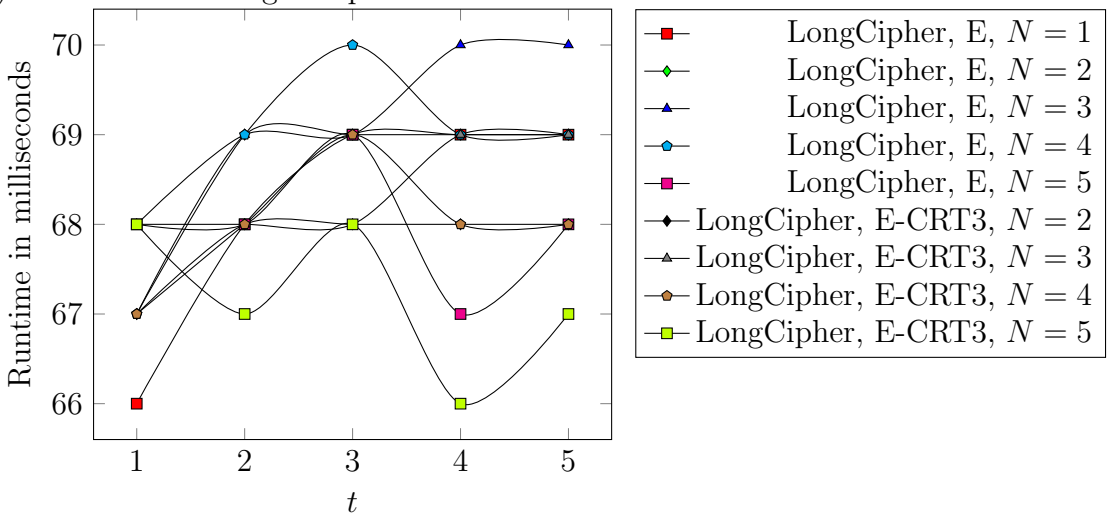
To understand how different parameters influence the runtimes of the schemes, we took a specific example where $bl(\Delta_{\mathbb{K}}) = 1828$ and $bl(f) = 256$. We plotted the runtimes in this instance and analyse the results in this and the next subsection.

Effect of t on Encryption and Decryption

Figure 5.1 – Figure 5.4 show the effect of increasing the t values for constant N values in the encryption process for full length and short length exponents, respectively. We immediately see that ShortCipher performs much better than LongCipher. This is due to the fact that, from Table 3.2, ShortCipher encryption performs 2 exponentiations in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ compared to LongCipher which performs the 2 exponentiations in a subgroup of $C(\mathcal{O}_{\Delta_f})$ where the integers representing the group elements are bigger in size compared to $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. Also, with increasing t, we see that the runtime tends to increase after $t = 1$ but remains fairly uniform thereafter. This can be attributed to the fact that when $t > 1$, the computation of $\mathfrak{f}^m$ is no longer simply computing the multiplicative inverse of the exponent m modulo f. Instead, we now use the interleaving wNAF method. Also, the use of E-CRT3 does seem to improve the runtime of the encryption process but this improvement is comparable to the encryption process used when $(N, t) = (1, 1)$. When the exponents are shorter in length, *i.e.*, of length 256 bits in this instance, we see that LongCipher performs better than ShortCipher. This is because from Table 3.2, LongCipher performs 2 exponentiations where exponents are of length 256 bits but ShortCipher performs 3 exponentiations where 2 of the exponents are of length 256 bits and 1 is of length $bl(f) = 256$ bits. Finally, note that in Figure 5.3 and Figure 5.4 the curves overlap since the runtimes remain largely the same for different N values.

Figure 5.5 – Figure 5.8 show the effect of increasing the t values for constant N values in the decryption process for full length and short length exponents. For full length exponents, we see that ShortCipher performs better than LongCipher in terms of overall de-

Figure 5.1: LongCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents



cryption runtimes. Again, this is because from Table 3.2, ShortCipher decryption performs 1 exponentiation in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ compared to LongCipher which performs 1 exponentiation in a subgroup of $C(\mathcal{O}_{\Delta_f})$ where the integers representing the group elements are bigger in size compared to $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. Also, with increasing t, we see that the runtimes increase after $t = 1$ when the Pohlig-Hellman subroutine is used in decryption. The use of D-CRT3 does seem to improve the runtime of the decryption process but this improvement is comparable to the decryption when $(N, t) = (1, 1)$. When the exponents are shorter in length, *i.e.*, of length 256 bits in this instance, we see that LongCipher performs better than ShortCipher in terms of decryption. This is because from Table 3.2, LongCipher performs 1 exponentiation where the exponent is of length 256 bits but ShortCipher performs 2 exponentiations where 1 of the exponents is of length 256 bits and the other is of length $bl(f)$.

Effect of N on Encryption and Decryption

Figure 5.9 – Figure 5.12 show the effect of increasing the N values for constant t values in the encryption process for full length and short length exponents.

We immediately see that ShortCipher performs much better than LongCipher. This is due to the fact that, from Table 3.2, ShortCipher encryption performs 2 exponentiations

Figure 5.2: ShortCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

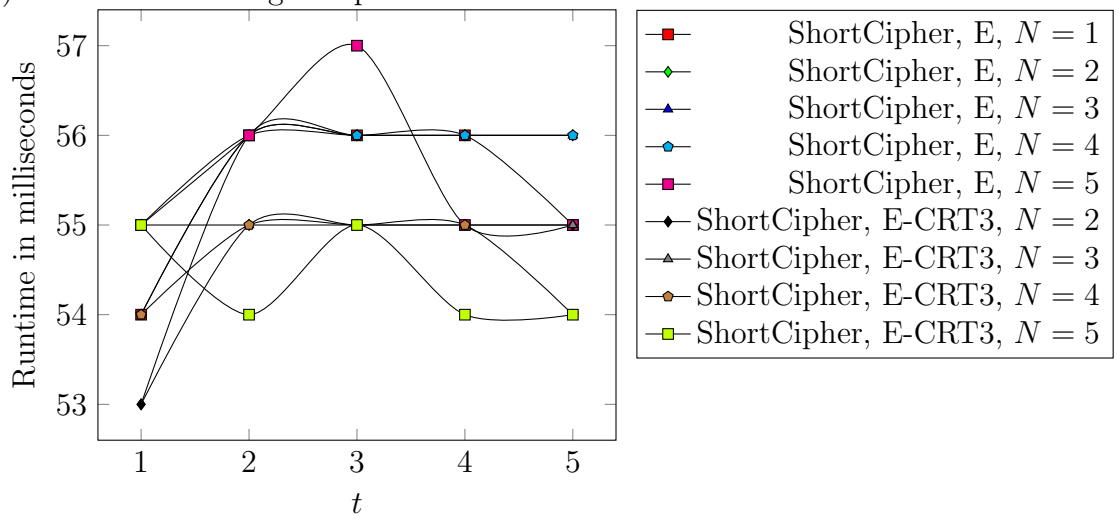


Figure 5.3: LongCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

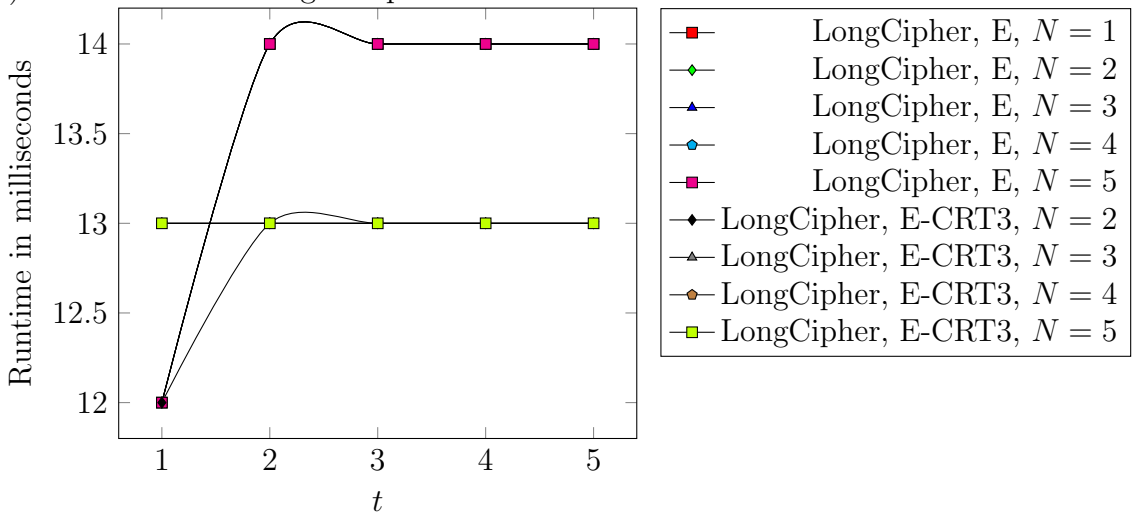


Figure 5.4: ShortCipher encryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

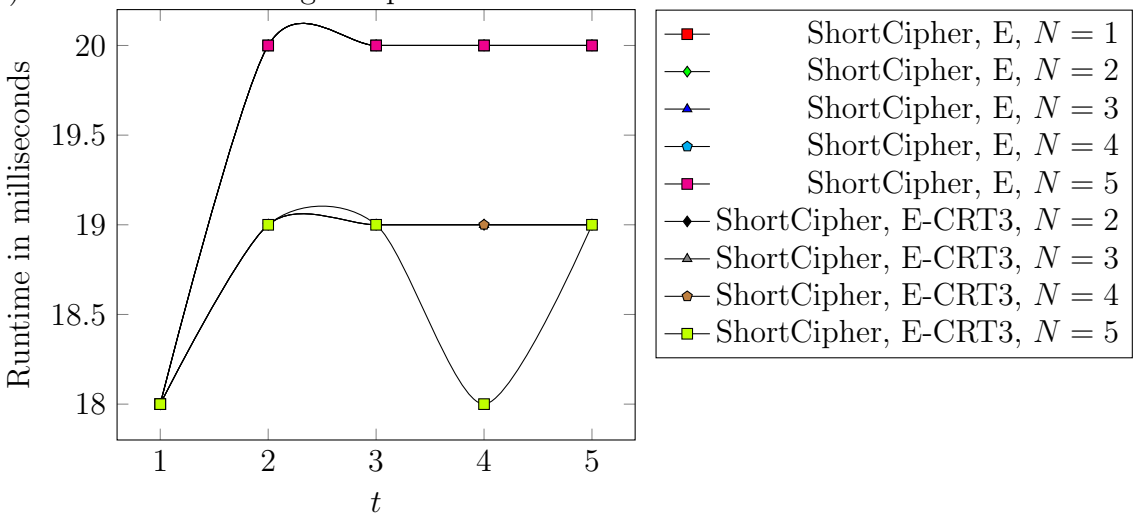


Figure 5.5: LongCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

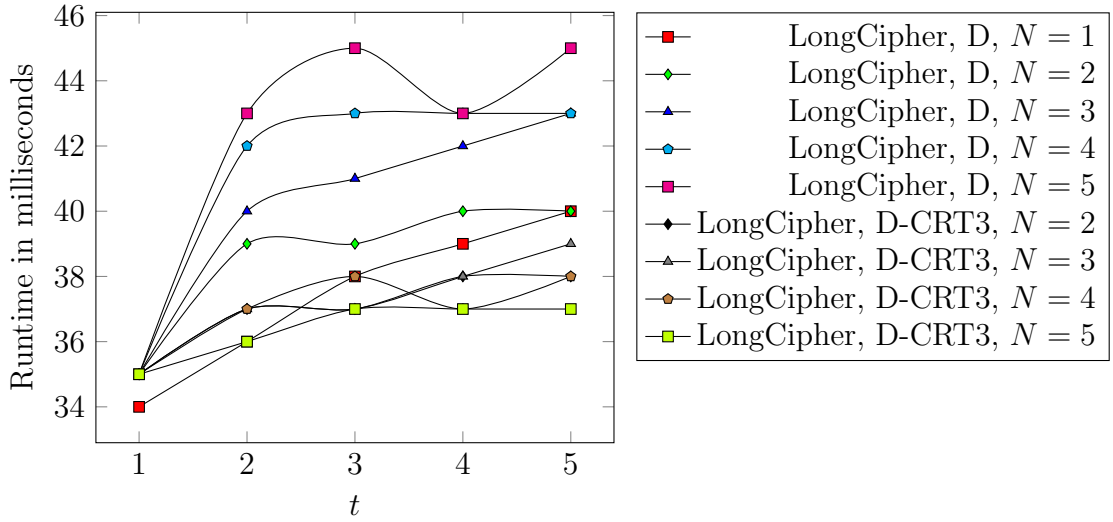


Figure 5.6: ShortCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

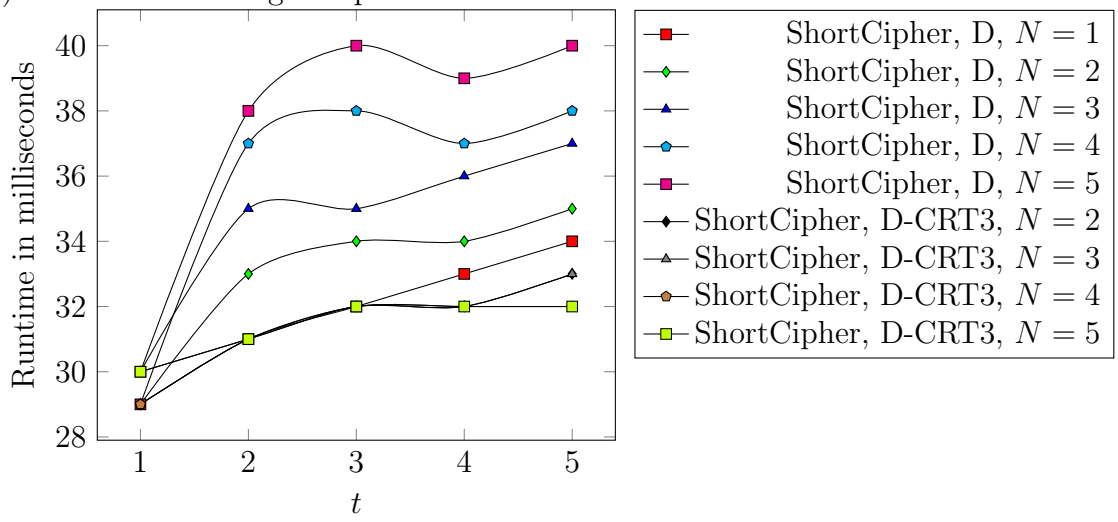


Figure 5.7: LongCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

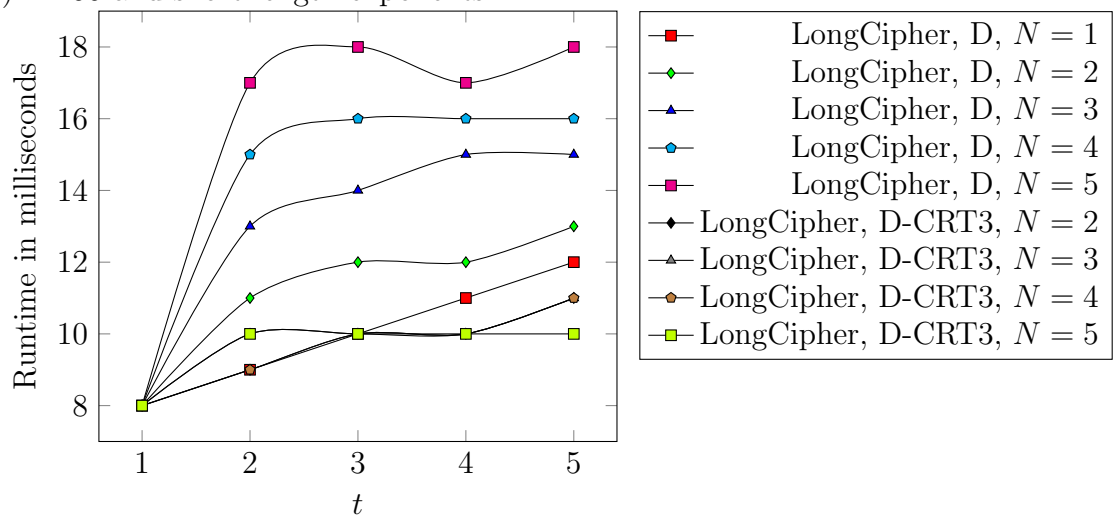
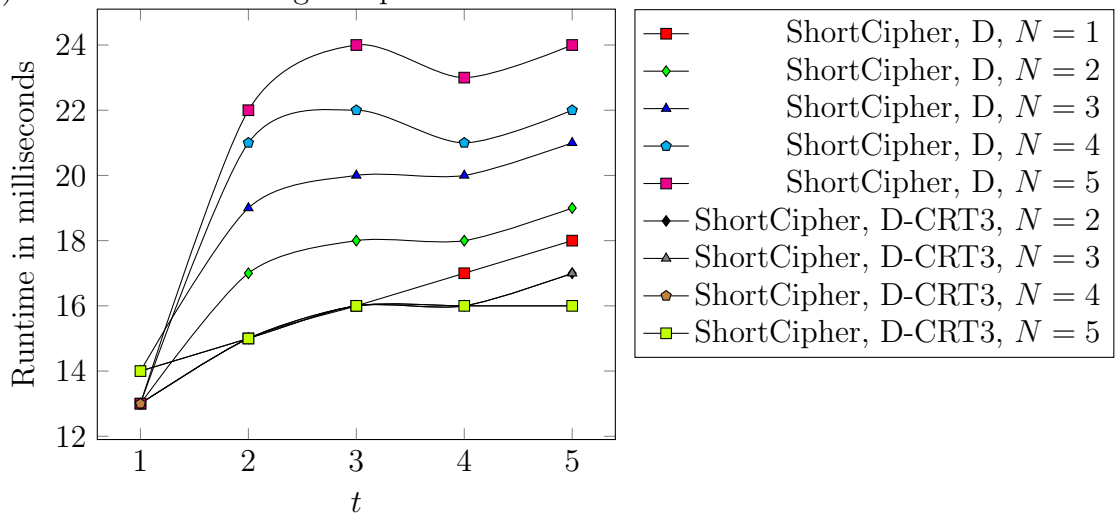


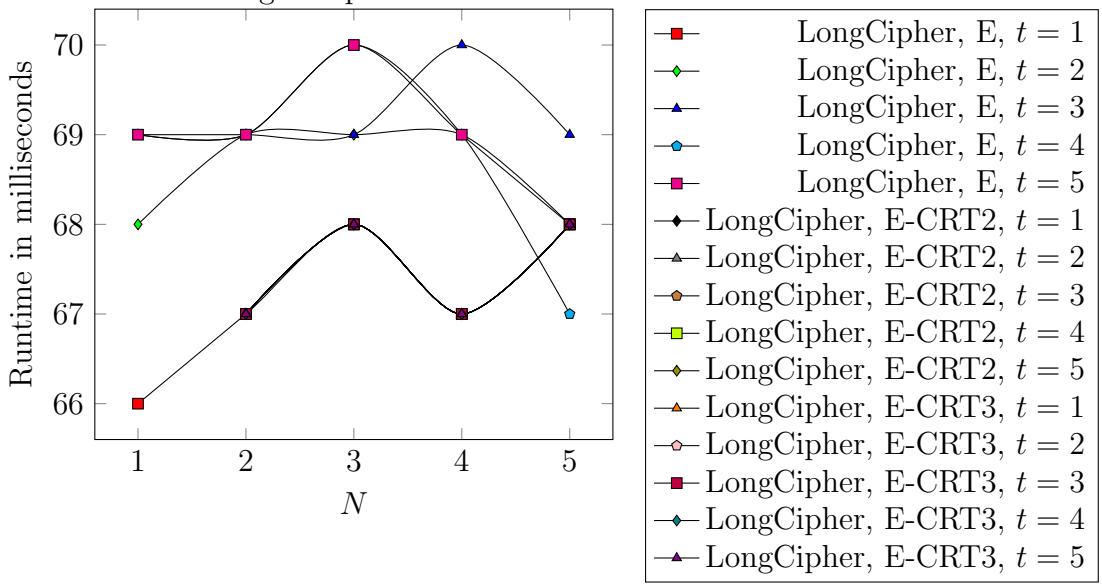
Figure 5.8: ShortCipher decryption runtimes with varying t, constant N, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents



in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ compared to LongCipher which performs the 2 exponentiations in a subgroup of $C(\mathcal{O}_{\Delta_f})$ whose elements are bigger in size compared to those of $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. Also, with increasing N and $t = 1$, we see that the runtime tends to remain more or less the same as computing $\mathfrak{f}^m$ only costs one inversion modulo f. When $t > 1$, we see that the runtimes of encryption tend to gradually increase. This can be attributed to the fact that the computation of $\mathfrak{f}^m$ is no longer simply computing the multiplicative inverse of the exponent m modulo f, but rather requires resorting to the interleaving method. The use of E-CRT3 does seem to improve the runtime but does not beat the runtime of $(N, t) = (1, 1)$. When the exponents are shorter in length, *i.e.*, of length 256 bits in this instance, we see that LongCipher performs better than ShortCipher. This is because from Table 3.2, LongCipher performs 2 exponentiations where exponents are of length 256 bits but ShortCipher performs 3 exponentiations where 2 of the exponents are of length 256 bits and 1 is of length $bl(f)$. Also, note that the runtime of the CRT schemes E-CRT2 and E-CRT3 largely remain the same across different (N, t) combinations.

Figure 5.13 – Figure 5.16 show the effect of increasing N values for constant t values in the decryption process for full length and short length exponents. For full length exponents, we see that ShortCipher performs better than LongCipher in terms of overall decryption

Figure 5.9: LongCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents



runtimes. Again, this is because from Table 3.2, ShortCipher decryption performs 1 exponentiation in $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$ compared to LongCipher which performs 1 exponentiation in a subgroup of $C(\mathcal{O}_{\Delta_f})$ where the integers representing the group elements are bigger in size compared to $C(\mathcal{O}_{\Delta_{\mathbb{K}}})$. With increasing N values, we see that the time to decrypt increases for each t value. Also, the use of D-CRT2 and D-CRT3 does seem to improve upon the runtime of the decryption process but this improvement is comparable to the decryption when $(N, t) = (1, 1)$. When the exponents are shorter in length, *i.e.*, of length 256 bits in this instance, we see that LongCipher performs better than ShortCipher in terms of decryption. This is because from Table 3.2, LongCipher performs 1 exponentiation where the exponent is of length 256 bits but ShortCipher performs 2 exponentiations where 1 of the exponents is of length 256 bits and the other is of length $bl(f)$. Finally, note that some of the curves in Figure 5.9 – Figure 5.16 overlap when the runtimes remain the same for different t values.

Figure 5.10: ShortCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

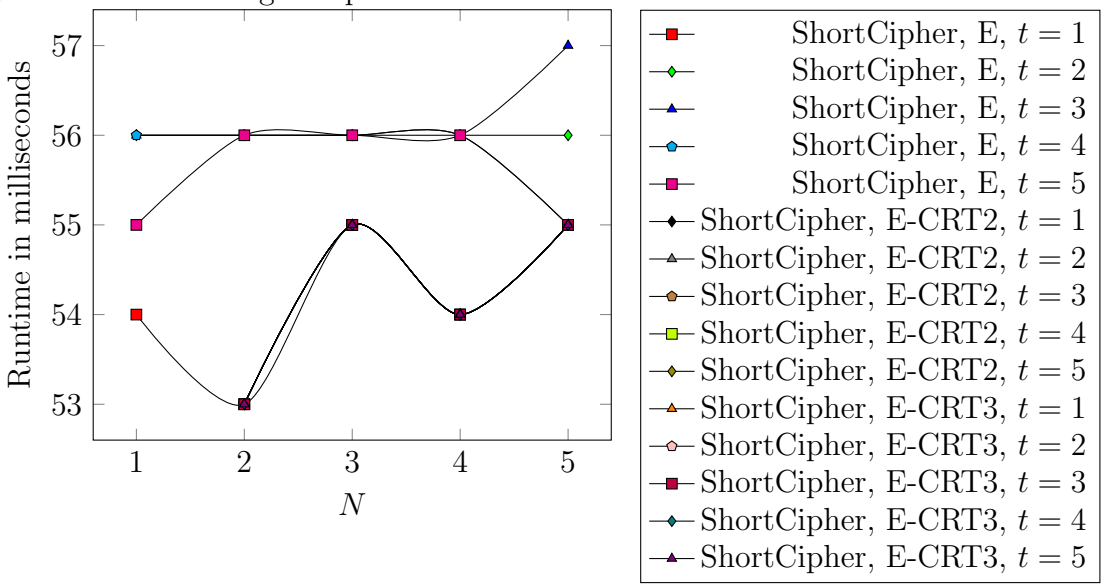


Figure 5.11: LongCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

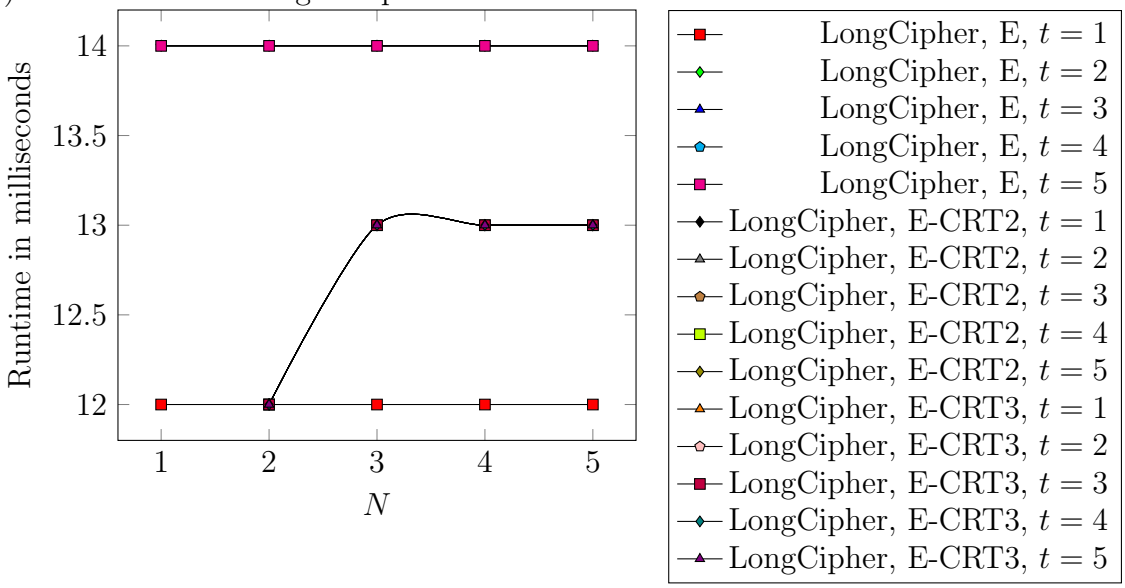


Figure 5.12: ShortCipher encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

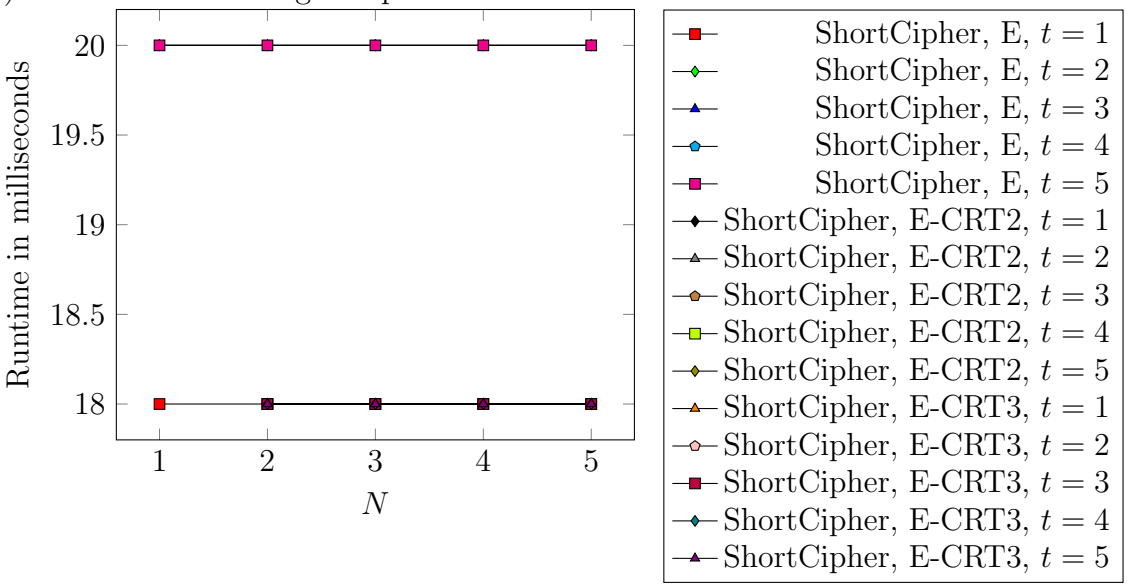


Figure 5.13: LongCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

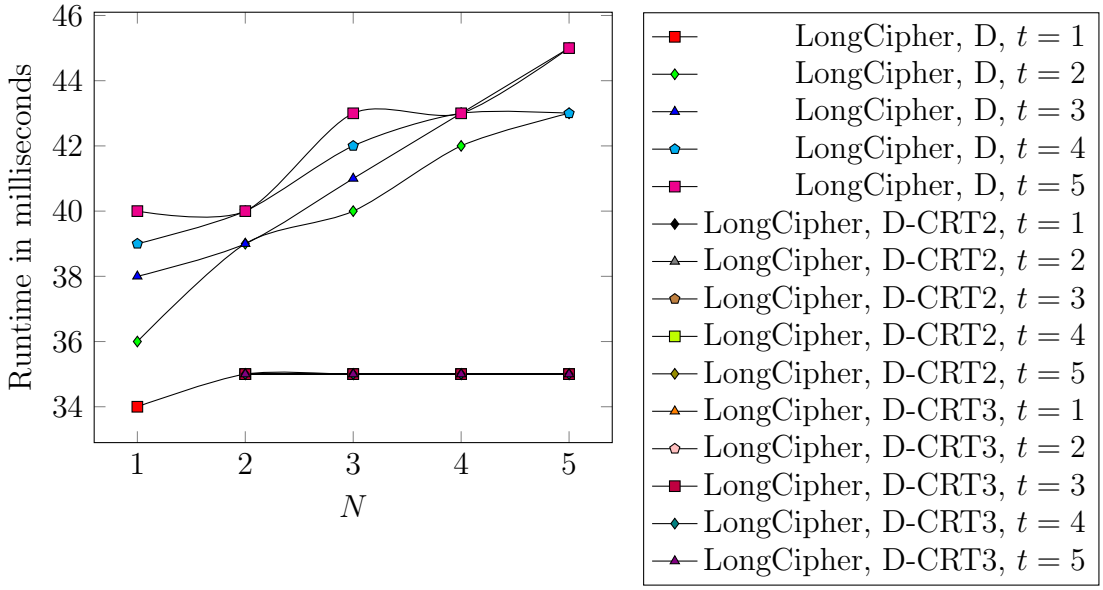


Figure 5.14: ShortCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and full length exponents

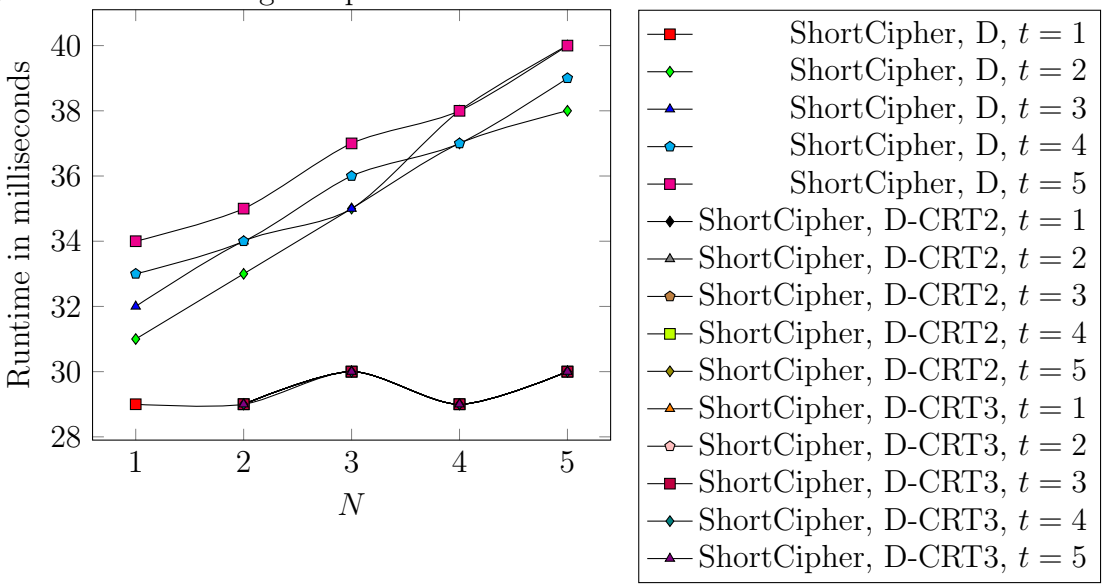


Figure 5.15: LongCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents

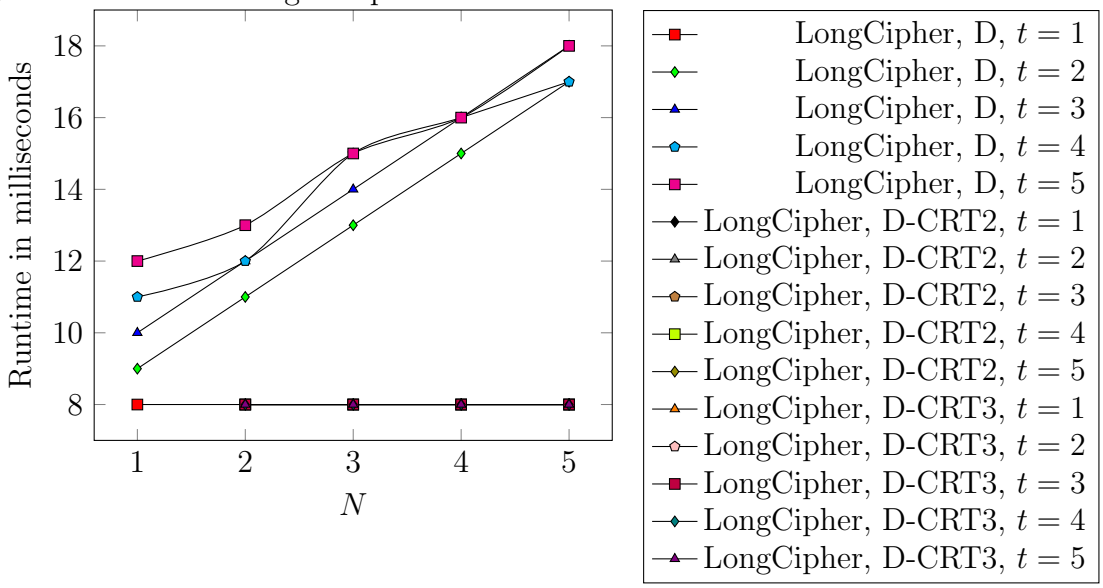
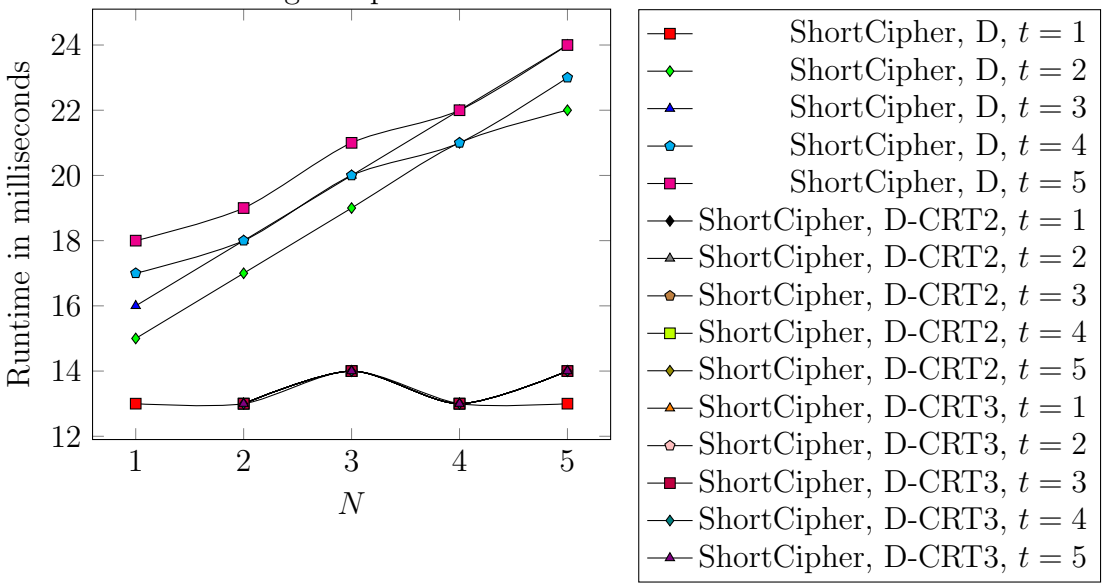


Figure 5.16: ShortCipher decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 256$ and short length exponents



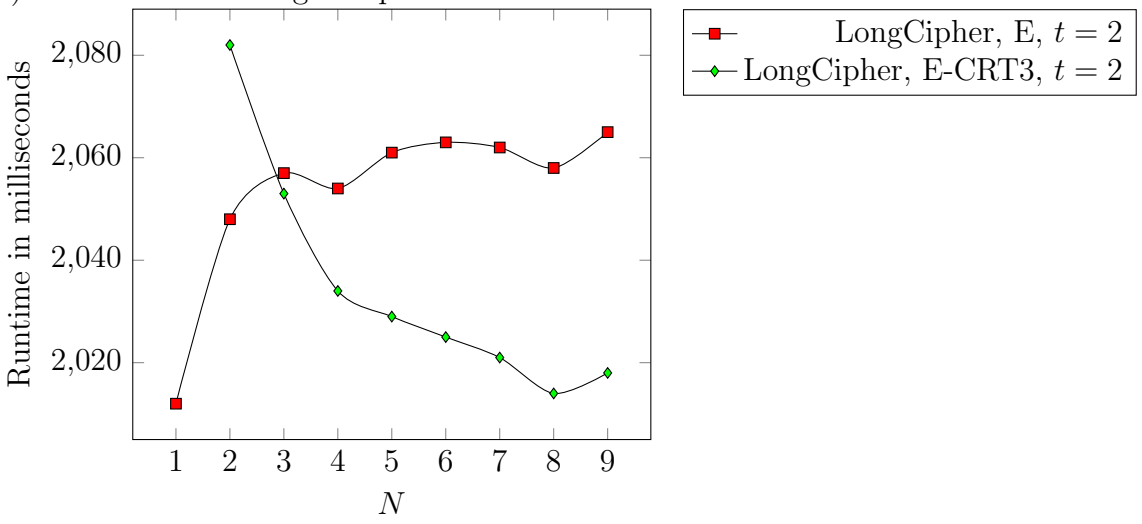
5.2.2 Case $bl(f) > bl(f)_{\max}$

Consider the tables in Appendix C and Appendix D when this is the case. One can clearly see that the encryption and decryption runtimes steadily climb with increasing N while E-CRT3 shows a steady improvement with increasing N. As a result, we only timed E-CRT3 and D-CRT3 beyond $N = 9$. Now, consider $bl(f) = 3072$ and $bl(\Delta_{\mathbb{K}}) = 1828$. Here, $t = \lceil bl(f)/bl(|\Delta_{\mathbb{K}}|) \rceil = \lceil 3072/1828 \rceil = 2$.

Effect of N on Encryption and Decryption

Figure 5.17 presents the effect of increasing the N values on LongCipher encryption for $t = 2$ while Figure 5.18 presents the same for ShortCipher. We see that ShortCipher has better performance than LongCipher and this is due to the smaller random exponents. This can be seen from Figure 5.21. When $bl(f) > bl(|\Delta_{\mathbb{K}}|)$, we see that the use of E-CRT3 improves the standard encryption technique in both LongCipher and ShortCipher. Figure 5.19 presents the effect of increasing the N values on LongCipher decryption for $t = 2$ while Figure 5.20 presents the same for ShortCipher. The use of Pohlig-Hellman in decryption results in a

Figure 5.17: LongCipher Encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents



steady climb of the runtime of the decryption function in both the schemes while the use of E-CRT3 immensely improves upon these runtimes. In Figure 5.21, we compare the E-CRT3 methods of LongCipher and ShortCipher and we see that the runtimes of both the schemes remain fairly consistent above $N = 9$ but ShortCipher in general provides a significant speedup over LongCipher.

5.2.3 Summary

In Table 5.4 and Table 5.5, we summarise these results. For each message size at each security level, we list the best encryption and decryption times, the type of encryption and decryption procedures and the corresponding (N, t) values that result in these times. We cross out the non-best entry if the best encryption and decryption runtimes arise from different variants. We record this for both full length and short length exponents. We explain the headers and the symbols used in Table 5.4 and Table 5.5 in Table 5.3.

As discussed above, we see that conductors with multiple prime divisors ($N > 1$) only improve performance for sufficiently large messages and large exponents. Using prime powers ($t > 1$) does not generally improve performance, but is necessary to handle messages that are

Figure 5.18: ShortCipher Encryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents

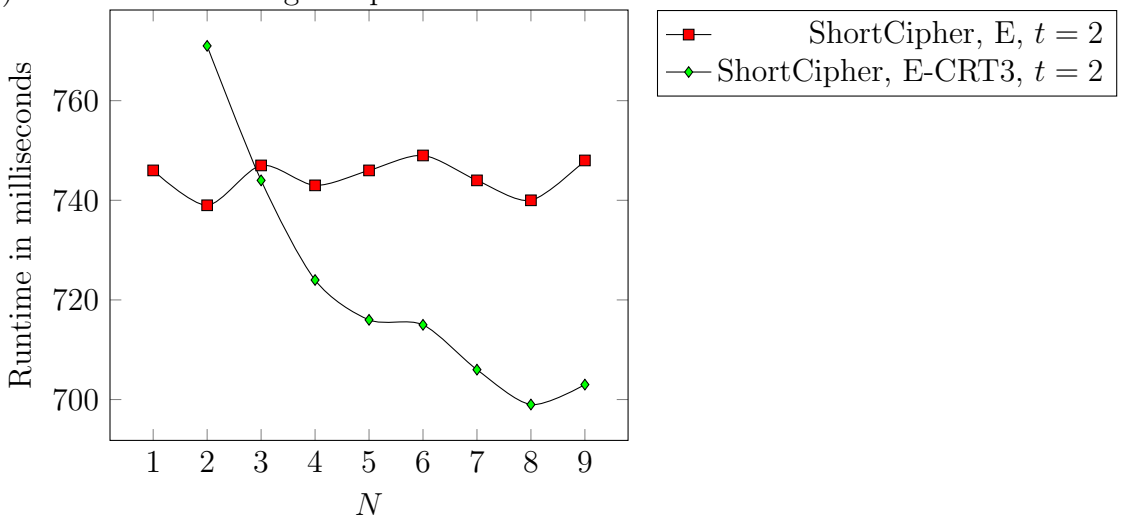


Figure 5.19: LongCipher Decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents

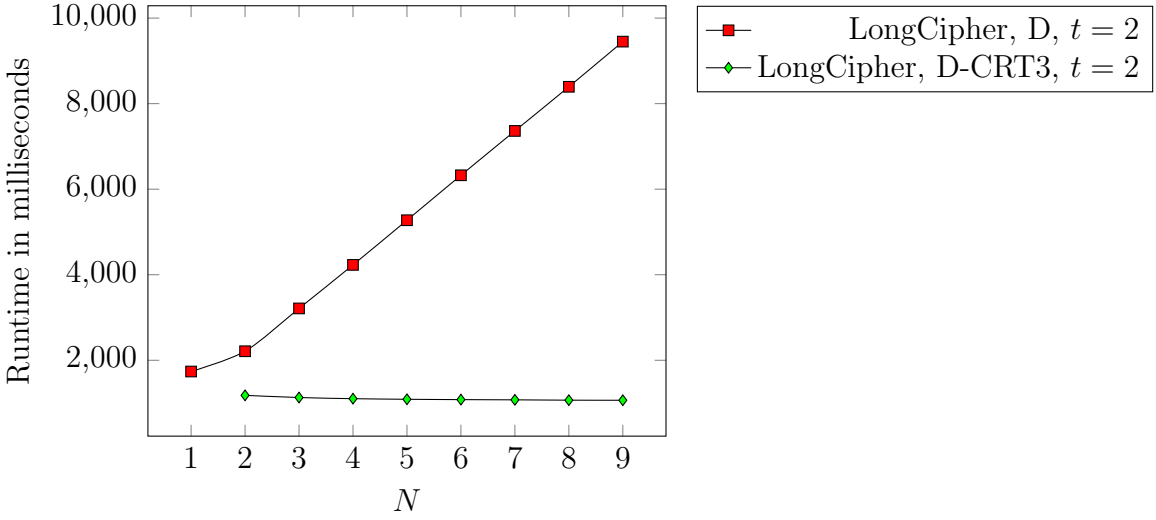


Figure 5.20: ShortCipher Decryption runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents

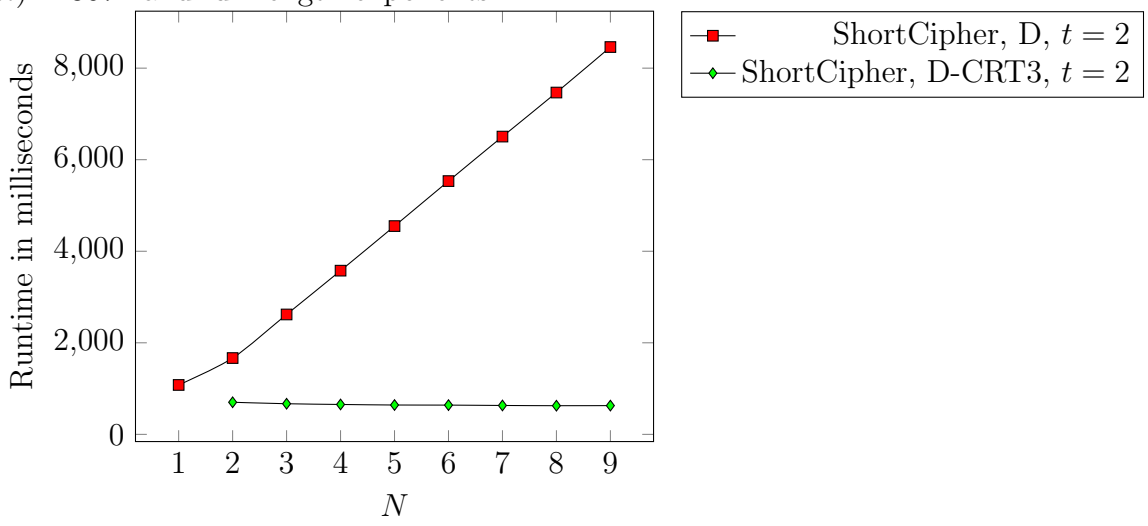


Figure 5.21: LongCipher and ShortCipher Encryption-CRT 3 runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents

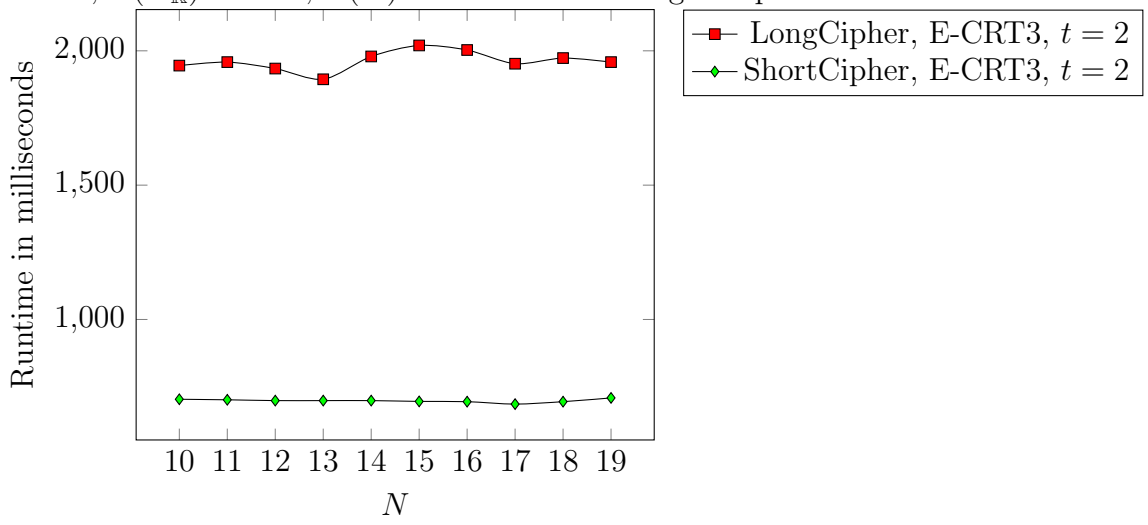
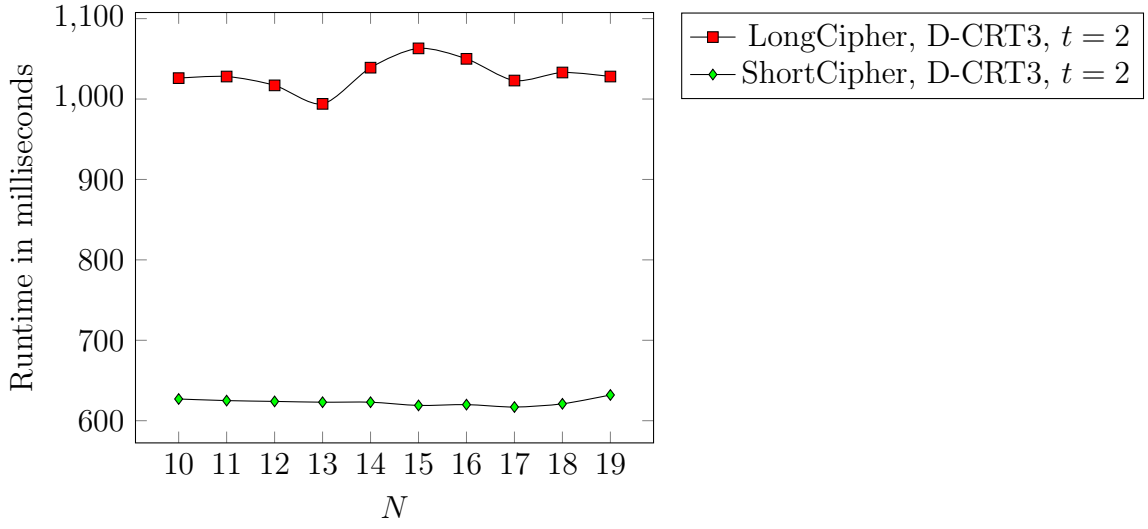


Figure 5.22: LongCipher and ShortCipher Decryption-CRT 3 runtimes with varying N, constant t, $bl(\Delta_{\mathbb{K}}) = 1828$, $bl(m) = 3072$ and full length exponents



larger than the fundamental discriminant. In that case, the smallest required value of t was optimal. Among the two schemes, LongCipher provides better results and is optimal when using short exponents, while ShortCipher provides better results with increasing message lengths with increasing security levels. Note that in Table 5.5, two different variants provide better results for encryption and decryption. In this case, we present the results of both the variants and strike out the slower results. When the message lengths were larger than the length of the fundamental discriminant, *i.e.*, when $bl(f) > bl(|\Delta_{\mathbb{K}}|)$, the CRT modification 3 significantly improves the runtimes.

Table 5.3: Description of headers and labels in Table 5.4 and Table 5.5

Security	Level of security in bits
Message	Bit length of the message being encrypted
N, t	N and t values of the conductor $f = (p_1 p_2 \cdots p_N)^t$
Scheme	CL scheme with the type of Encrypt and Decrypt functions
Encryption	Encryption runtime in milliseconds
Decryption	Decryption runtime in milliseconds
Exponent length	$2b$ bits in Table 5.4 where b is the security level in bits
	At most the bit length of Bf (from Algorithm 4.2.1) in Table 5.5
LC, SC	LongCipher, ShortCipher respectively
ED	Encryption and Decryption from Section 3.3
ED-CRT3	CRT 3 Encryption and Decryption from Section 4.3

Table 5.4: Summary of the CL cryptosystem with short exponent lengths (in ms)

Security	Message	N, t	Scheme	Encryption	Decryption
128	16	$1, 1$	LC-ED	9	6
	80	$1, 1$	LC-ED	10	6
	256	$1, 1$	LC-ED	12	8
	3072	$10, 2$	LC-ED-CRT3	99	99
192	16	$1, 1$	LC-ED	31	18
	80	$1, 1$	LC-ED	33	19
	256	$1, 1$	LC-ED	37	21
	7680	$20, 3$	LC-ED-CRT3	711	911
256	16	$1, 1$	LC-ED	79	44
	80	$1, 1$	LC-ED	81	45
	256	$1, 1$	LC-ED	88	49
	15360	$20, 3$	LC-ED-CRT3	2235	2946

Table 5.5: Summary of the CL cryptosystem with full exponent lengths(in ms)

Security	Message	N, t	Scheme	Encryption	Decryption
128	16	$1, 1$	LC-ED	36	19
	80	$1, 1$	SC-ED	43	22
	256	$1, 1$	SC-ED	54	29
	3072	$17, 2$	SC-ED-CRT3	685	617
192	16	$1, 1$	LC-ED	151	78
	80	$1, 1$	LC-ED	167	86
	256	$1, 1$	SC-ED	205	98
	7680	$19, 3$	SC-ED-CRT3	7488	7230
256	16	$1, 1$	LC-ED	467	238
	80	$1, 1$	LC-ED	504	256
	256	$1, 1$	LC-ED	568	~~289~~
		$1, 1$	SC-ED	~~601~~	270
	15360	$20, 3$	SC-ED-CRT3	27155	25931

5.3 Comparison of CL, Paillier and BCP schemes

We next compare the best versions of the Castagnos and Laguillaumie cryptosystem to the Paillier [35] and Bresson *et al.* [8] schemes. As stated in the previous section, we implemented the BCP and Paillier schemes. Since the BCP scheme is based on both the hardness of the integer factorisation problem and the discrete logarithm problem, we have two versions of the BCP scheme as well, one with full length exponents and the other with short length exponents. Note that contrary to the BCP scheme, the Paillier scheme encryption performs operations with the message as an exponent. Thus, for a fixed security level, we expect that Paillier encryption times for message lengths 16, 80 and 256 bits will vary from messages of length 3072 bits whereas the other operations should remain relatively constant.

We compare below these results with those of the best results from the Castagnos and Laguillaumie variants in Table 5.6, Table 5.7 and Table 5.8. For the Castagnos and Laguillaumie timings, we list the best observed encryption and decryption times amongst all the variants we implemented.

Table 5.6: Summary of best performance (in ms) — 128-bit security

		Short Exponents		Full Exponents	
Message	System	Encryption	Decryption	Encryption	Decryption
16	CL	9	6	36	19
	BCP	4	2	79	40
	Paillier			21	7
80	CL	10	6	43	22
	BCP	4	2	79	40
	Paillier			21	7
256	CL	12	8	53	29
	BCP	4	2	79	40
	Paillier			21	7
3072	CL	99	99	685	617
	BCP	4	2	79	40
	Paillier			42	7

In Figure 5.23 and Figure 5.24, we plot the encryption and decryption runtimes of the

Table 5.7: Summary of best performance (in ms) — 192-bit security

		Short Exponents		Full Exponents	
Message	System	Encryption	Decryption	Encryption	Decryption
16	CL	31	18	151	78
	BCP	21	10	827	414
	Paillier			214	74
80	CL	31	19	167	86
	BCP	21	10	827	414
	Paillier			214	74
256	CL	37	21	205	98
	BCP	21	10	827	414
	Paillier			214	74
7680	CL	711	911	7488	7230
	BCP	21	10	827	414
	Paillier			424	74

Table 5.8: Summary of best performance (in ms) — 256-bit security

		Short Exponents		Full Exponents	
Message	System	Encryption	Decryption	Encryption	Decryption
16	CL	79	44	446	227
	BCP	85	44	4838	2417
	Paillier			1252	428
80	CL	81	45	475	242
	BCP	85	44	4838	2417
	Paillier			1252	428
256	CL	88	49	568	~~289~~
				~~601~~	270
	BCP	85	44	4838	2417
	Paillier			1252	428
15360	CL	2235	2946	27155	25931
	BCP	85	44	4838	2417
	Paillier			2484	428

three schemes at the 128-bit security level. In Figure 5.25 and Figure 5.26, we plot the same for the 192-bit security level. Finally, in Figure 5.27 and Figure 5.28, we plot the same for the 256-bit security level.

At the 128-bit security, Paillier was the fastest for both encryption and decryption when using full length exponents and BCP was the fastest for short exponents, for all message lengths considered. BCP was fastest for short exponents at the 192-bit level, while the Castagnos and Laguillaumie variants were superior when using full length exponents for 16- and 80-bit messages; the results were mixed for larger messages. At the 256-bit security level, Castagnos and Laguillaumie variants are fastest for the three smallest message sizes when using full length and short length exponents. Among the Castagnos and Laguillaumie schemes under consideration here, the ShortCipher with $(N, t) = (1, 1)$ proved to be the best for full length exponents while LongCipher with $(N, t) = (1, 1)$ proved to be the best for short exponents. Other variations and conductor decompositions were advantageous once the messages and exponents were larger than $bl(|\Delta_{\mathbb{K}}|)$.

Figure 5.23: Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 128-bit security level

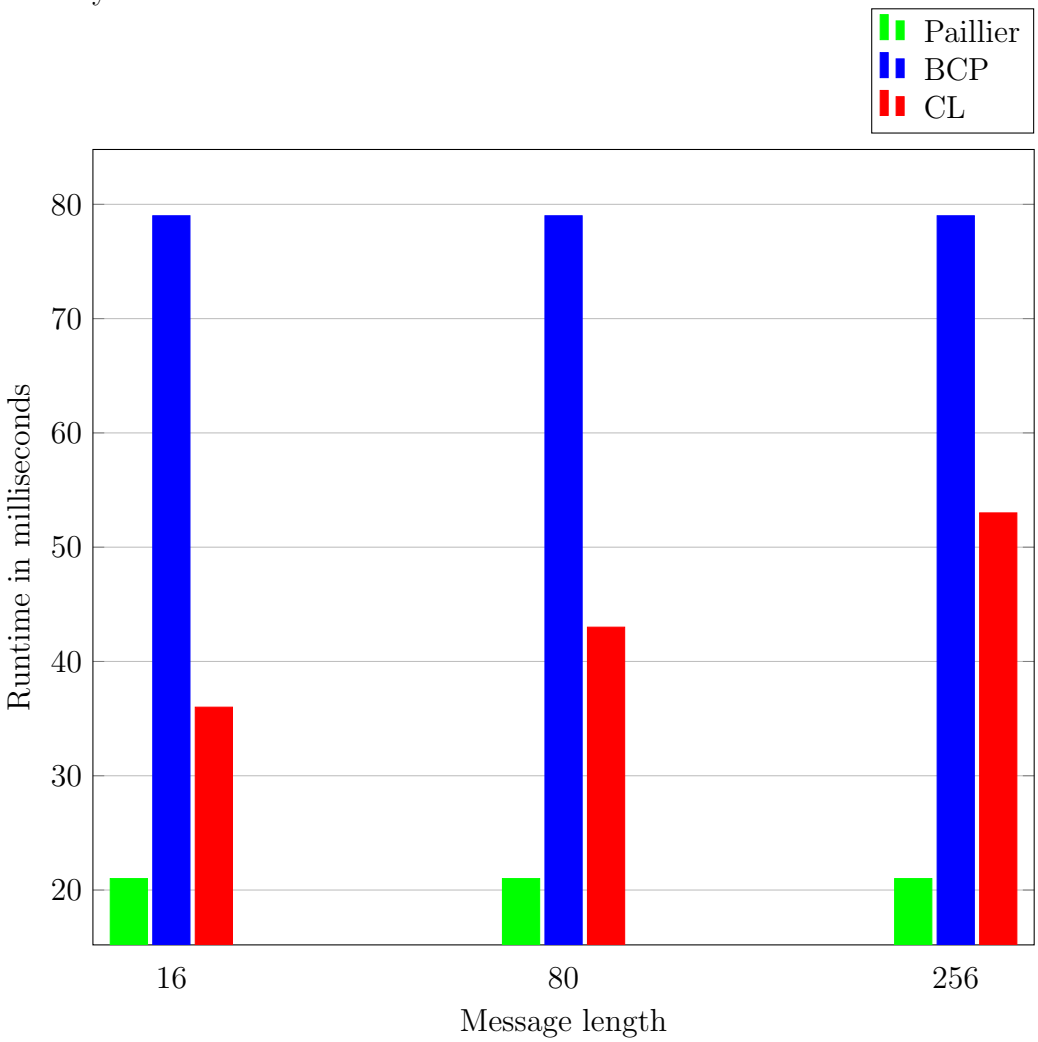


Figure 5.24: Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 128-bit security level

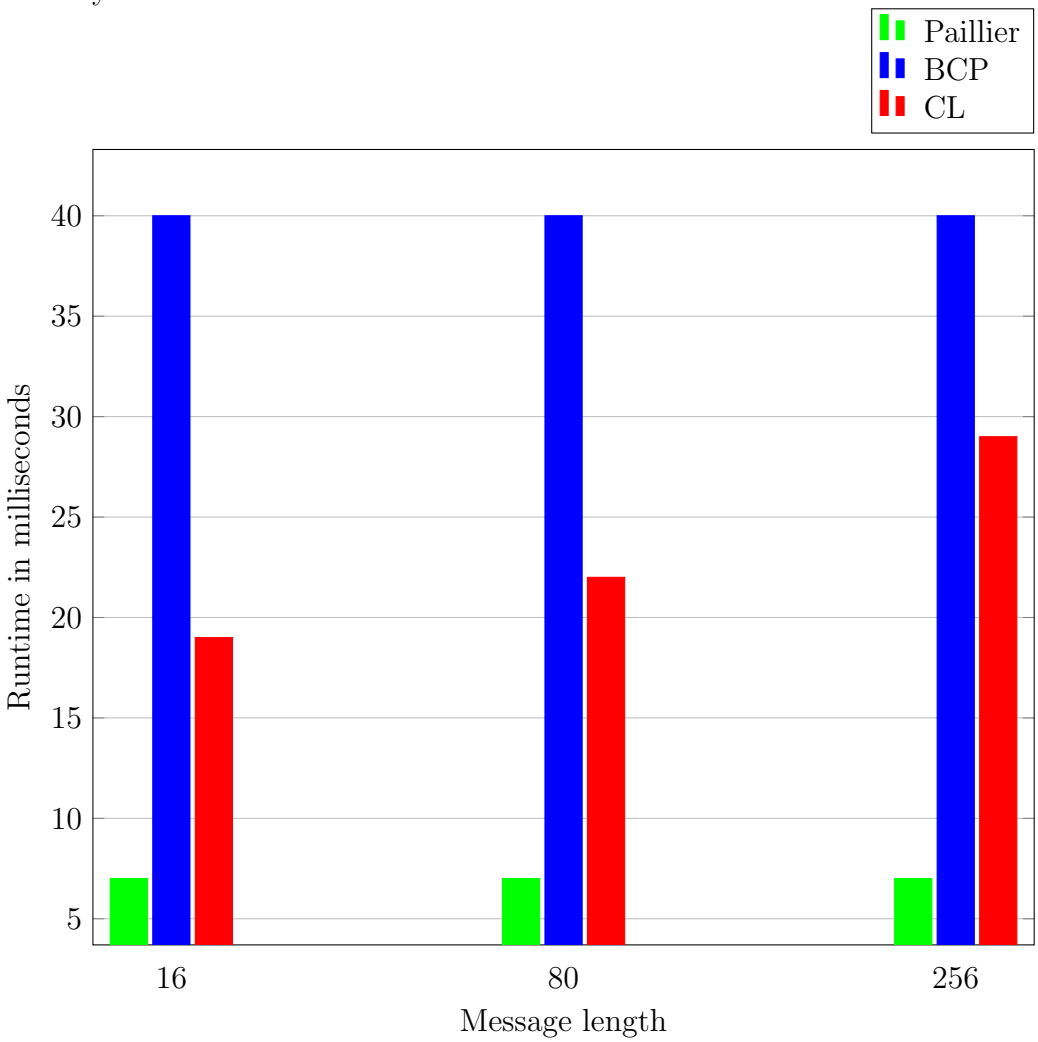


Figure 5.25: Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 192-bit security level

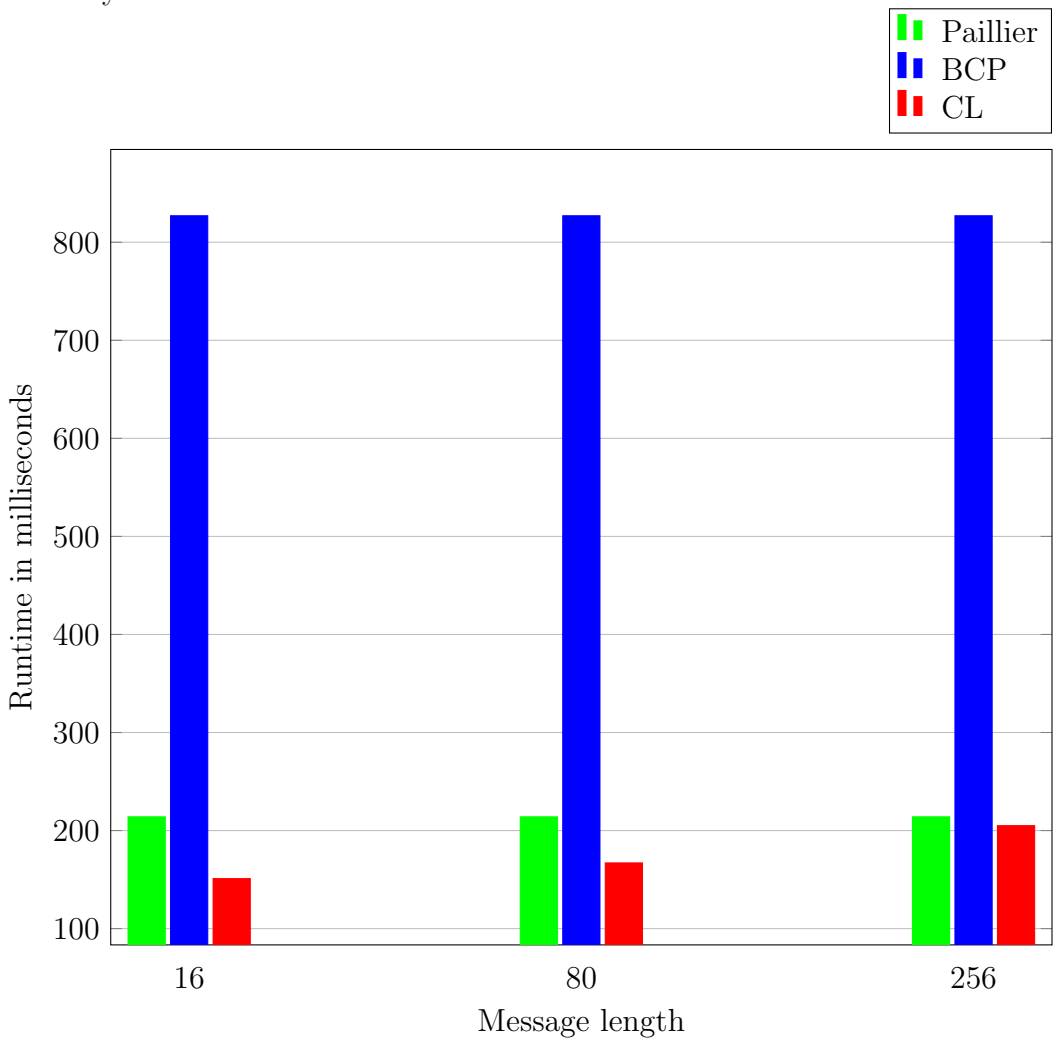


Figure 5.26: Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 192-bit security level

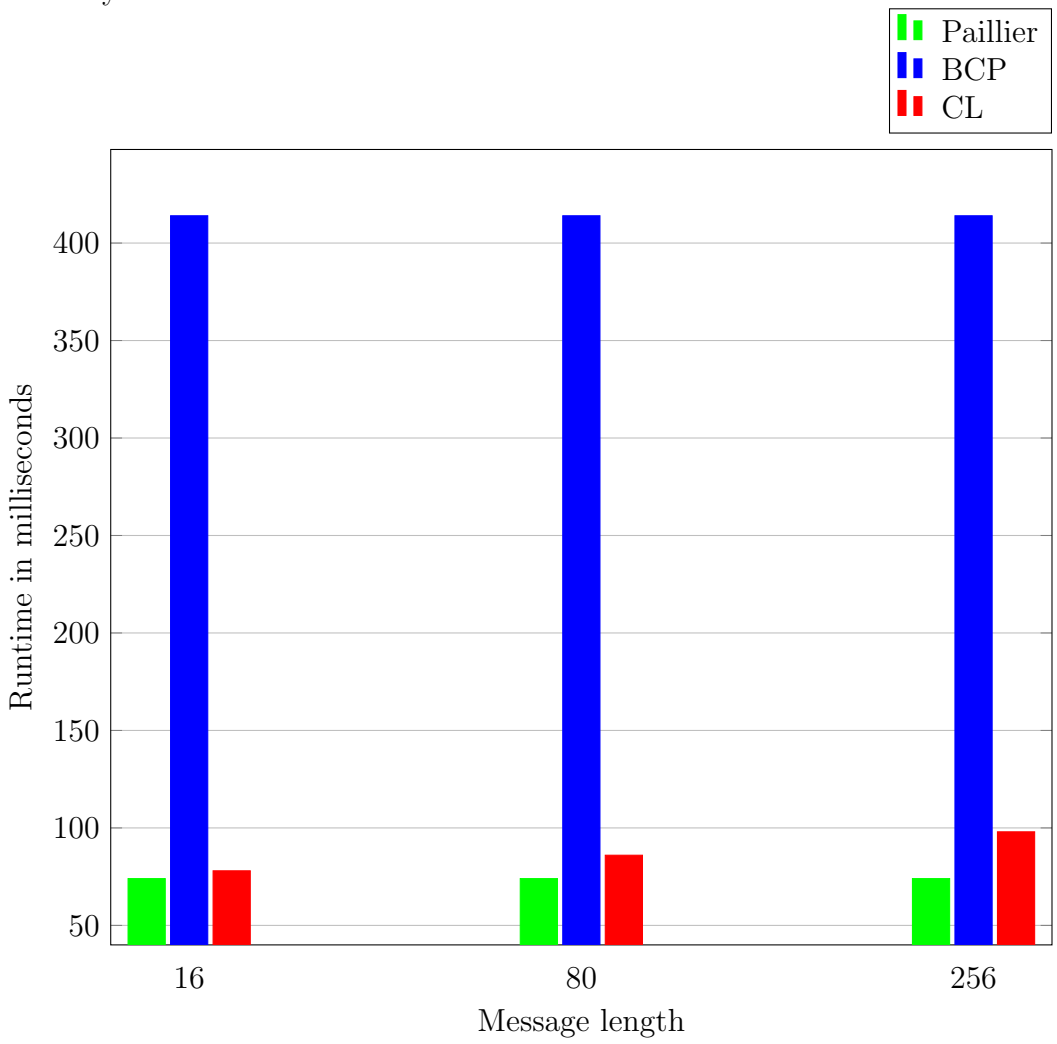


Figure 5.27: Comparison of encryption runtimes of Paillier, BCP and CL schemes at the 256-bit security level

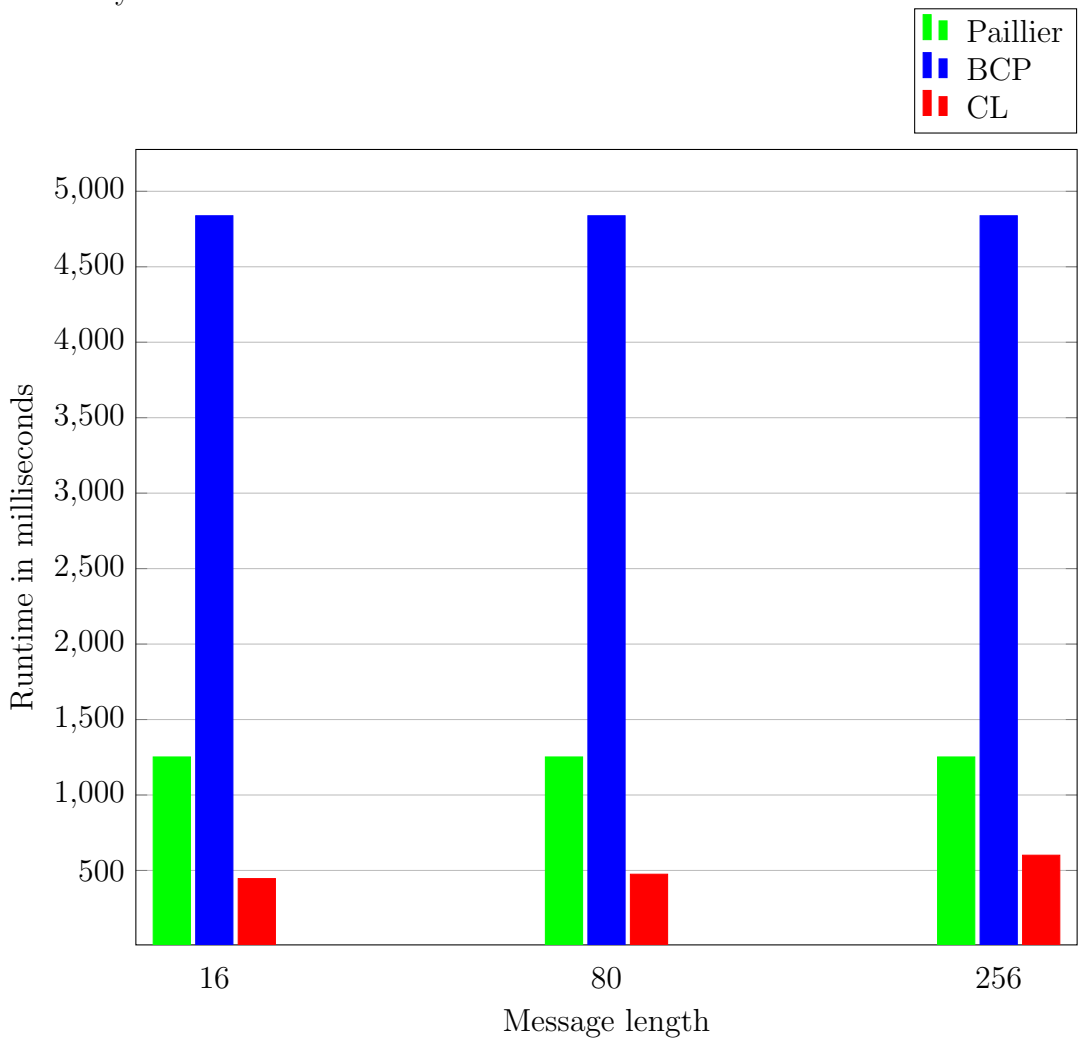
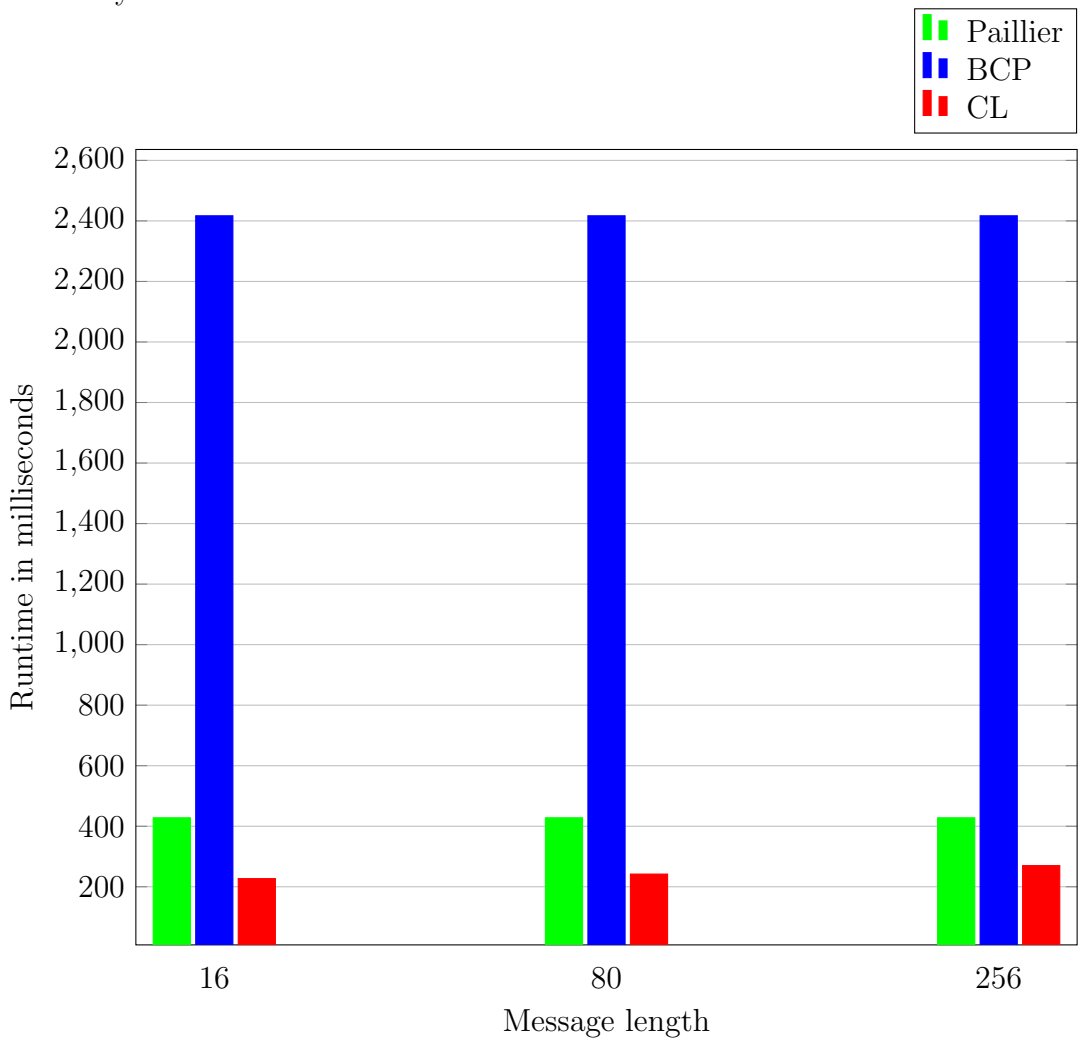


Figure 5.28: Comparison of decryption runtimes of Paillier, BCP and CL schemes at the 256-bit security level



Chapter 6

Conclusion

In this thesis, we looked in detail at a linearly homomorphic encryption scheme by Castagnos and Laguillaumie which is based on class groups of orders of imaginary quadratic fields. Castagnos and Laguillaumie used a prime conductor $f = p$ to set up two schemes, referred to as LongCipher and ShortCipher, in these orders and we extended their schemes using a non prime conductor $f = (p_1 p_2 \cdots p_N)^t$. Some of these extensions improved the performance of the CL schemes. When compared with the Paillier and the BCP schemes, the CL schemes outperformed both in specific cases.

Our experiments show that of the two schemes presented by Castagnos and Laguillaumie, ShortCipher works best with full length exponents and for larger message lengths at a given security level while LongCipher works best with short exponent lengths irrespective of message length. Prime conductors provide the best result for small messages while we showed that conductors of form $f = (p_1 p_2 \cdots p_N)^t$ $N, t > 1$ provide a way to to encrypt large messages using the Castagnos and Laguillaumie setup. We chose conductors of form $f = p_1^t p_2^t \cdots p_N^t$ instead of the generic form $f' = p_1^{t_1} p_2^{t_2} \cdots p_N^{t_N}$, where $t_1, t_2, \ldots t_N$ are non-negative integers, in order to simplify the generation of conductors. One can also use conductors of form f' but this does not improve the efficiency in any way as conductors of form $f = p_1^t p_2^t \cdots p_N^t$ are simply used to increase the size of the message used without increasing the size of the

fundamental discriminant. The CRT modifications presented in Chapter 4 improve upon the runtimes when $f = (p_1 p_2 \cdots p_N)^t$ and $N, t > 1$.

For smaller message lengths, the CL encryption scheme outperform BCP and Paillier at higher security levels with full length exponents and BCP performs the best with short length exponents. With full length exponents, CL outperforms BCP and Paillier with 16, 80 and 256 bit messages at 256 bit security level. With full length exponents, Paillier provides the best results with 16,80 and 256bit messages at lower levels of security and when 3072, 7680 and 15360 bit messages are considered.

6.1 Future Work

We discuss some open problems arising from our work that warrant further investigation. In Chapter 4, we presented extensions and several optimisations to the Castagnos and Laguillaumie cryptosystem. One further optimisation that could be considered to improve the extended versions is to use word size architecture instead of multi-precision for sufficiently small prime divisors of the conductor. This was not done in our experiments and could potentially make these versions more competitive.

Other projects include testing the CL schemes against schemes based on Learning With Errors (LWE) [38] and Elgamal Elliptic Curve based schemes [29]. One could also implement the Paillier scheme improvements presented in [27] and benchmark the CL schemes and its variants against these improved variants.

In [13], Castagnos *et al.* presented a functional encryption scheme using their idea of (p^2, p) being a reduced ideal in $\mathcal{O}_{\Delta_p}$ *i.e.*, using a conductor $f = p$. In [10], Catagnos *et al.* used the same idea to present an efficient encryption switching protocol. A nice application of our work is to extend the schemes presented in [13] and [10] using conductor $f = (p_1 p_2 \cdots p_N)^t$ and potentially improve the performance of the systems.

When $bl(f) > bl(|\Delta_{\mathbb{K}}|)$ and $f = (p_1 p_2 \cdots p_N)^t$, it would be interesting to check the

existence of a subgroup $\mathcal{F} \subset C(\mathcal{O}_{\Delta_f})$ of order p_i^t where $\mathcal{F} = \langle \mathfrak{f} \rangle$, $\mathfrak{f} = [(p_i^{2t}, p_i^t)]$ and $\mathfrak{f}^m = [(p_i^{2t}, L(m)p_i^t)]$ where $mL(m) \equiv 1 \pmod{p_i^t}$. If such a group exists, then the Castagnos and Laguillaumie schemes would provide much better results for large messages that we considered.

We remark that the short exponent versions of the cryptosystems, as expected, are quite efficient. It would be of interest to revise and complete the security proofs in this context, where the intractability assumptions are all replaced by their short exponent analogues.

Bibliography

[1] Jithra Adikari, Vassil S. Dimitrov, and Laurent Imbert. Hybrid Binary-Ternary Joint Sparse Form and its Application in Elliptic Curve Cryptography. *IACR Cryptology ePrint Archive*, 2008:285, 2008.

[2] Şaban Alaca and Kenneth S. Williams. *Introductory Algebraic Number Theory*. Cambridge University Press, Cambridge, 2004.

[3] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A Guide to Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2015:1192, 2015.

[4] Elaine B. Barker. *Recommendation for Key Management, Part 1: General*. NIST Special Publication (NIST SP) - 800-57 Part 1 Revision 4. NIST Pubs, 2016.

[5] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[6] Jean-François Biasse, Michael J. Jacobson, Jr., and Alan K. Silvester. Security Estimates for Quadratic Field Based Cryptosystems. In *Information Security and Privacy - 15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Pro-*

ceedings, volume 6168 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2010.

[7] Wieb Bosma and Peter Stevenhagen. On the Computation of Quadratic 2-Class Groups. *Journal de Théorie des Nombres de Bordeaux*, 8(2):283–313, 1996.

[8] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and its Applications. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[9] Johannes Buchmann and Ulrich Vollmer. *Binary Quadratic Forms*, volume 20 of *Algorithms and Computation in Mathematics*. Springer, Berlin, 2007. An algorithmic approach.

[10] Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. Encryption Switching Protocols Revisited: Switching Modulo p. In *Advances in Cryptology—CRYPTO 2017. Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 255–287. Springer, Cham, 2017.

[11] Guilhem Castagnos and Fabien Laguillaumie. On the Security of Cryptosystems with Quadratic Decryption: The Nicest Cryptanalysis. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 260–277. Springer, 2009.

[12] Guilhem Castagnos and Fabien Laguillaumie. Linearly Homomorphic Encryption from DDH. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA*

Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505. Springer, 2015.

[13] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, pages 733–764, 2018.

[14] Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[15] Henri Cohen and Hendrik W. Lenstra, Jr. Heuristics on Class Groups of Number Fields. In *Number theory, Noordwijkerhout 1983 (Noordwijkerhout, 1983)*, volume 1068 of *Lecture Notes in Math.*, pages 33–62. Springer, Berlin, 1984.

[16] Harvey Cohn. *Advanced Number Theory*. Dover Books on Mathematics. Dover Publications, 1980.

[17] David A. Cox. *Primes of the Form $x^2 + ny^2$*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs, and Tracts. John Wiley & Sons, Inc., New Jersey, second edition, 2013.

[18] Parthasarathi Das, Michael J. Jacobson Jr., and Renate Scheidler. Improved Efficiency of a Linearly Homomorphic Cryptosystem. In *Codes, Cryptology and Information Security - Third International Conference, C2SI 2019, Rabat, Morocco, April 22-24, 2019, Proceedings - In Honor of Said El Hajji*, volume 11445 of *Lecture Notes in Computer Science*, pages 349–368. Springer, 2019.

[19] David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley & Sons, Inc., Hoboken, NJ, third edition, 2004.

[20] Taher Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. Inform. Theory*, 31(4):469–472, 1985.

[21] GMP. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/. [Online; accessed 31-May-2016].

[22] James L. Hafner and Kevin S. McCurley. A Rigorous Subexponential Algorithm for Computation of Class Groups. *J. Amer. Math. Soc.*, 2(4):837–850, 1989.

[23] Safuat Hamdy and Bodo Möller. Security of Cryptosystems Based on Class Groups of Imaginary Quadratic Orders. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 2000.

[24] Detlef Hühnlein, Michael J. Jacobson, Jr., Sachar Paulus, and Tsuyoshi Takagi. A Cryptosystem Based on Non-Maximal Imaginary Quadratic Orders with Fast Decryption. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 1998.

[25] Michael J. Jacobson, Jr. Computing Discrete Logarithms in Quadratic Orders. *J. Cryptology*, 13(4):473–492, 2000.

[26] Michael J. Jacobson, Jr. and Hugh C. Williams. *Solving the Pell Equation*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC. Springer, New York, 2009.

[27] Christine Jost, Ha Lam, Alexander Maximov, and Ben J. M. Smeets. Encryption Performance Improvements of the Paillier Cryptosystem. *IACR Cryptology ePrint Archive*, 2015:864, 2015.

[28] Pierre Kaplan. Divisibility by 8 of the Number of Classes of Quadratic Bodies whose 2-Class Group is Cyclic, and Reciprocal Biquadratic. *J. Math., Soc., Japan*, 25(4):596–608, 10 1973.

[29] Neal Koblitz. Elliptic Curve Cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[30] Takeshi Koshiba and Kaoru Kurosawa. Short Exponent Diffie-Hellman Problems. In *Public key cryptography—PKC 2004*, volume 2947 of *Lecture Notes in Comput. Sci.*, pages 173–186. Springer, Berlin, 2004.

[31] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[32] Bodo Möller. Algorithms for Multi-Exponentiation. In *Selected areas in cryptography*, volume 2259 of *Lecture Notes in Comput. Sci.*, pages 165–180. Springer, Berlin, 2001.

[33] Bodo Möller. Improved Techniques for Fast Exponentiation. In *Information security and cryptology—ICISC 2002*, volume 2587 of *Lecture Notes in Comput. Sci.*, pages 298–312. Springer, Berlin, 2003.

[34] Keith W. Nicholson. *Introduction to Abstract Algebra*. Wiley, fourth edition, 2012.

[35] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceedings*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[36] PARI/GP. PARI/GP - A Computer Algebra System for Fast Computations in Number Theory. https://pari.math.u-bordeaux.fr/. [Online; accessed 31-May-2016].

[37] Stephen C. Pohlig and Martin E. Hellman. An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. *IEEE Trans. Information Theory*, IT-24(1):106–110, 1978.

[38] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[39] George W. Reitwiesner. Binary arithmetic. In *Advances in Computers*, volume 1, pages 231–308. Elsevier, 1960.

[40] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978.

[41] SageMath. SageMath - Open-Source Mathematical Software System. http://www.sagemath.org/. [Online; accessed 31-May-2016].

[42] Maxwell Sayles. Improved Arithmetic in the Ideal Class Group of Imaginary Quadratic Number Fields With an Application to Integer Factoring. Master's thesis, University of Calgary, May 2013.

[43] Maxwell Sayles. `liboptarith` and `libqform` repositories — GitHub repository. https://github.com/maxwellsayles/, 2014. [Online; accessed 31-May-2016].

[44] Daniel Shanks. Class Number, A Theory of Factorization and Genera. In *Proceedings of Symposia in Pure Mathematics*, volume 20, pages 415–440, Providence, Rhode Island, 1971. American Mathematical Society.

[45] Daniel Shanks. On Gauss and Composition. I, II. In *Number Theory and Applications (Banff, AB, 1988)*, volume 265 of *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, pages 163–178, 179–204. Kluwer Acad. Publ., Dordrecht, 1989.

[46] Victor Shoup. NTL: A Library for Doing Number Theory. https://www.shoup.net/ntl/. [Online; accessed 31-May-2016].

[47] Jerome A. Solinas. Low-Weight Binary Representations for Pairs of Integers. Technical report, University of Waterloo, 2001.

Appendix A

BCP Runtimes

In Table A.1, we present the results of our implementation of the BCP scheme. Column 1 displays the security level in bits from Table 3.1. Column 2 displays the bit lengths of the messages being encrypted at a given security level. Columns 3–6 present the encryption and decryption runtimes in milliseconds for short and full exponents where short exponent version is described in Subsection 4.4.2.

Table A.1: Performance of BCP scheme

Security	Message	Short Exponents		Full Exponents	
		Encryption	Decryption	Encryption	Decryption
128	16	4	2	79	40
	80	4	2	79	40
	256	4	2	79	40
	3072	4	2	79	40
192	16	21	10	827	414
	80	21	10	827	414
	256	21	10	827	414
	7680	21	10	827	414
256	16	85	44	4838	2417
	80	85	44	4838	2417
	256	85	44	4838	2417
	15360	85	44	4838	2417

Appendix B

Paillier Runtimes

In Table B.1, we present the results of our implementation of the Paillier scheme. Column 1 displays the security level in bits from Table 3.1. Column 2 displays the bit lengths of the messages being encrypted at a given security level. Columns 3,4 and 5 present the encryption and decryption and decryption using CRT runtimes in milliseconds.

Table B.1: Performance of Paillier scheme

Security	Message	Encrypt	Decrypt	Decrypt-CRT
128	16	21	21	7
	80	21	21	7
	256	21	21	7
	3072	42	21	7
192	16	214	214	74
	80	214	214	74
	256	214	214	74
	7680	424	214	74
256	16	1252	1252	428
	80	1252	1252	428
	256	1252	1252	428
	15360	2484	1252	428

Appendix C

CL Runtimes - LongCipher

In this appendix, we present the results of our implementation of the LongCipher scheme. The captions of each table describe the name of the CL scheme which is LongCipher in this chapter, followed by the bit length of the fundamental discriminant $bl(\Delta_{\mathbb{K}})$, bit length of a single message $bl(m)$, and the size of the random exponents which are either short or full length as described in Table 5.3. We describe the notation used in these results in the following table.

	$bl(m) < bl(\Delta_{\mathbb{K}})$	$bl(m) > bl(\Delta_{\mathbb{K}})$
N	number of primes in $f = (p_1 p_2 \cdots p_N)^t$	
t	exponent used in $f = (p_1 p_2 \cdots p_N)^t$	
E	Algorithm 3.2.2	Algorithm 3.2.2
D	Algorithm 3.2.3	Algorithm 4.3.1
D-CRT1	Algorithm 4.2.2	
E-CRT2	Algorithm 4.2.3	
D-CRT2	Algorithm 4.2.4	
E-CRT3	Algorithm 4.2.5	Algorithm 4.3.2
D-CRT3	Algorithm 4.2.6	Algorithm 4.3.3

Table C.1: LongCipher, 1828, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	9	6					
1	2	9	6					
2	1	9	6	6	9	6	9	6

Table C.2: LongCipher, 1828, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	36	19					
1	2	36	19					
2	1	36	19	19	36	19	36	19

Table C.3: LongCipher, 1828, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	10	6					
1	2	11	7					
1	3	11	7					
1	4	11	7					
1	5	11	7					
2	1	10	6	6	10	6	10	6
2	2	11	7				10	7
2	3	11	8				11	7
2	4	11	8				11	7
2	5	11	8				11	7
3	1	11	7	7	11	7	11	7
3	2	11	8				11	7
4	1	11	6	6	11	6	11	7
4	2	11	8				11	7
5	1	11	7	6	11	7	11	7

Table C.4: LongCipher, 1828, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	44	23					
1	2	44	24					
1	3	45	24					
1	4	44	24					
1	5	44	24					
2	1	44	23	23	44	23	44	23
2	2	44	24				44	23
2	3	44	24				44	24
2	4	44	24				44	24
2	5	45	24				44	24
3	1	43	23	23	43	23	43	23
3	2	44	24				44	24
4	1	43	23	23	44	23	44	23
4	2	44	25				44	24
5	1	43	23	23	44	23	44	23

Table C.5: LongCipher, 1828, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	12	8					
1	2	14	9					
1	3	14	10					
1	4	14	11					
1	5	14	12					
2	1	12	8	8	12	8	12	8
2	2	14	11				13	9
2	3	14	12				13	10
2	4	14	12				13	10
2	5	14	13				13	11
3	1	12	8	8	13	8	13	8
3	2	14	13				13	10
3	3	14	14				13	10
3	4	14	15				13	10
3	5	14	15				13	11
4	1	12	8	8	13	8	13	8
4	2	14	15				13	9
4	3	14	16				13	10
4	4	14	16				13	10
4	5	14	16				13	11
5	1	12	8	8	13	8	13	8
5	2	14	17				13	10
5	3	14	18				13	10
5	4	14	17				13	10
5	5	14	18				13	10

Table C.6: LongCipher, 1828, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	66	34					
1	2	68	36					
1	3	69	38					
1	4	69	39					
1	5	69	40					
2	1	67	35	35	67	35	67	35
2	2	69	39				68	37
2	3	69	39				68	37
2	4	69	40				68	38
2	5	69	40				68	38
3	1	68	35	35	68	35	68	35
3	2	69	40				68	37
3	3	69	41				68	37
3	4	70	42				69	38
3	5	70	43				69	39
4	1	67	35	35	67	35	67	35
4	2	69	42				68	37
4	3	70	43				69	38
4	4	69	43				68	37
4	5	69	43				68	38
5	1	68	35	35	68	35	68	35
5	2	68	43				67	36
5	3	69	45				68	37
5	4	67	43				66	37
5	5	68	45				67	37

Table C.7: LongCipher, 1828, 3072, short length

N	t	E	D	E-CRT3	D-CRT3
1	2	471	799		
2	2	481	1255	186	231
3	2	486	2263	151	177
4	2	485	3274	133	151
5	2	486	4317	124	135
6	2	487	5367	118	126
7	2	488	6412	114	121
8	2	486	7441	110	115
9	2	489	8506	108	111
10	2			99	100
11	2			98	98
12	2			97	96
13	2			97	96
14	2			97	96
15	2			96	92
16	2			95	92
17	2			95	92
18	2			96	93
19	2			96	93
20	2			93	88
21	2			94	89
22	2			94	89
23	2			94	89
24	2			91	86

Table C.8: LongCipher, 1828, 3072, full length

N	t	E	D	E-CRT3	D-CRT3
1	2	2012	1737		
2	2	2048	2211	2082	1180
3	2	2057	3212	2053	1129
4	2	2054	4230	2034	1102
5	2	2061	5274	2029	1088
6	2	2063	6325	2025	1080
7	2	2062	7362	2021	1075
8	2	2058	8394	2014	1067
9	2	2065	9450	2018	1065
10	2			1945	1026
11	2			1958	1028
12	2			1934	1017
13	2			1894	994
14	2			1979	1039
15	2			2020	1063
16	2			2003	1050
17	2			1952	1023
18	2			1973	1033
19	2			1958	1028
20	2			1984	1039
21	2			1860	975
22	2			1878	983
23	2			1879	983
24	2			1841	963

Table C.9: LongCipher, 3598, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	31	18					
1	2	32	18					
2	1	31	18	18	32	18	32	18

Table C.10: LongCipher, 3598, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	151	78					
1	2	151	78					
2	1	151	78	78	151	78	151	78

Table C.11: LongCipher, 3598, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	33	19					
1	2	34	20					
1	3	34	20					
1	4	34	20					
1	5	34	20					
2	1	33	19	19	33	19	33	19
2	2	34	20				33	20
2	3	34	20				33	20
2	4	34	20				33	20
2	5	34	21				33	20
3	1	33	19	19	33	19	33	19
3	2	34	21				33	20
4	1	33	19	19	33	19	33	19
4	2	34	21				34	20
5	1	33	19	19	33	19	33	19

Table C.12: LongCipher, 3598, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	161	83					
1	2	166	85					
1	3	164	84					
1	4	164	85					
1	5	164	85					
2	1	163	84	84	163	84	164	84
2	2	164	85				163	85
2	3	164	85				162	84
2	4	163	85				162	84
2	5	165	86				164	85
3	1	162	83	83	162	83	162	83
3	2	164	85				163	85
4	1	162	84	84	163	84	163	84
4	2	165	86				164	85
5	1	162	83	83	162	83	162	83

Table C.13: LongCipher, 3598, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	37	21					
1	2	42	24					
1	3	42	25					
1	4	42	26					
1	5	41	27					
2	1	37	21	21	37	21	37	21
2	2	42	26				38	23
2	3	42	26				38	24
2	4	42	27				38	24
2	5	42	28				38	26
3	1	37	21	21	37	21	37	21
3	2	42	28				38	24
3	3	41	29				38	24
3	4	41	29				38	24
3	5	42	30				38	25
4	1	37	21	21	38	22	38	22
4	2	41	30				38	23
4	3	42	31				38	24
4	4	42	31				38	24
4	5	42	31				38	25
5	1	37	21	21	38	21	38	22
5	2	42	32				38	24
5	3	42	34				38	24
5	4	42	33				38	24
5	5	42	34				38	25

Table C.14: LongCipher, 3598, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	216	111					
1	2	221	113					
1	3	220	114					
1	4	221	115					
1	5	220	116					
2	1	216	111	111	216	111	216	111
2	2	221	115				217	113
2	3	220	115				216	113
2	4	220	116				217	114
2	5	221	117				217	115
3	1	216	111	110	216	111	216	111
3	2	220	117				216	113
3	3	220	118				217	113
3	4	221	119				217	114
3	5	221	120				218	115
4	1	218	112	112	218	112	218	112
4	2	223	121				219	115
4	3	222	121				218	114
4	4	222	121				218	114
4	5	221	121				218	115
5	1	217	112	111	218	112	218	112
5	2	222	122				219	114
5	3	223	124				220	115
5	4	221	122				217	114
5	5	221	123				217	114

Table C.15: LongCipher, 3598, 7680, short length

N	t	E	D	E-CRT3	D-CRT3
1	3	5142	18699		
2	3	5165	18105	2062	4659
3	3	5137	29966	1529	3209
4	3	4875	40499	1218	2408
5	3	4835	52439	1065	1988
6	3	4835	64848	990	1777
7	3	4836	77404	934	1621
8	3	4819	89630	880	1468
9	3	4833	102246	838	1343
10	3			808	1254
11	3			788	1192
12	3			775	1148
13	3			758	1091
14	3			750	1064
15	3			737	1013
16	3			729	985
17	3			716	939
18	3			713	929
19	3			711	916
20	3			711	911
21	3			712	770
22	3			687	713
23	3			687	710
24	3			697	716

Table C.16: LongCipher, 3598, 7680, full length

N	t	E	D	E-CRT3	D-CRT3
1	3	21604	28963		
2	3	21283	27854	22012	14560
3	3	20292	37874	20497	12564
4	3	20199	49661	20173	11873
5	3	20237	62030	20068	11493
6	3	20236	74458	19992	11281
7	3	20239	87033	19939	11127
8	3	20172	99243	19831	10946
9	3	20239	111968	19844	10849
10	3			19742	10717
11	3			19693	10641
12	3			19562	10534
13	3			19539	10455
14	3			19276	10311
15	3			19200	10226
16	3			19162	10179
17	3			19125	10127
18	3			19119	10112
19	3			19140	10112
20	3			19163	10123
21	3			23164	11995
22	3			23121	11943
23	3			23144	11955
24	3			23499	12118

Table C.17: LongCipher, 5972, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	79	44					
1	2	80	45					
2	1	79	44	44	80	44	80	44

Table C.18: LongCipher, 5972, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	467	238					
1	2	468	239					
2	1	467	238	238	468	238	468	238

Table C.19: LongCipher, 5972, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	81	45					
1	2	84	46					
1	3	84	46					
1	4	84	46					
1	5	84	47					
2	1	81	45	45	82	45	82	45
2	2	85	47				82	46
2	3	85	47				82	47
2	4	85	47				82	47
2	5	86	48				84	48
3	1	81	45	45	82	45	82	46
3	2	84	47				82	46
4	1	81	45	45	82	45	82	45
4	2	85	48				83	47
5	1	81	45	45	83	45	83	46

Table C.20: LongCipher, 5972, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	504	256					
1	2	505	256					
1	3	507	257					
1	4	506	257					
1	5	506	257					
2	1	504	256	256	504	256	504	256
2	2	508	258				505	258
2	3	507	258				505	258
2	4	509	259				506	258
2	5	519	264				516	264
3	1	504	256	256	505	256	505	257
3	2	507	258				505	257
4	1	502	255	255	503	256	504	256
4	2	510	261				508	259
5	1	503	256	256	506	256	506	257

Table C.21: LongCipher, 5972, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	88	49					
1	2	97	51					
1	3	98	53					
1	4	97	54					
1	5	98	56					
2	1	88	49	49	88	49	88	49
2	2	98	54				89	51
2	3	98	55				89	52
2	4	97	56				89	53
2	5	98	57				89	54
3	1	88	49	49	89	49	89	49
3	2	98	57				90	52
3	3	98	58				89	52
3	4	98	59				90	53
3	5	98	60				90	54
4	1	88	49	49	89	49	89	49
4	2	97	59				89	51
4	3	98	61				90	53
4	4	97	60				89	52
4	5	97	61				90	54
5	1	88	49	49	90	49	90	49
5	2	97	62				90	52
5	3	98	64				90	52
5	4	98	63				90	53
5	5	98	64				90	53

Table C.22: LongCipher, 5972, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	568	289					
1	2	576	291					
1	3	580	294					
1	4	577	293					
1	5	579	296					
2	1	571	290	290	570	290	572	290
2	2	579	294				572	292
2	3	580	296				573	293
2	4	579	296				572	293
2	5	581	298				574	295
3	1	581	295	295	581	295	582	295
3	2	588	302				581	297
3	3	587	302				580	297
3	4	588	303				581	298
3	5	590	305				583	300
4	1	576	293	293	577	293	578	293
4	2	586	303				579	296
4	3	591	307				584	299
4	4	585	303				578	296
4	5	585	304				579	297
5	1	572	291	291	574	291	575	291
5	2	579	302				572	292
5	3	584	306				578	295
5	4	580	303				574	293
5	5	580	305				574	294

Table C.23: LongCipher, 5972, 15360, short length

N	t	E	D	E-CRT3	D-CRT3
1	3	22759	68652		
2	3	21470	62630	6916	16097
3	3	21338	103519	4945	10757
4	3	21343	147583	4042	8298
5	3	21322	192542	3552	6943
6	3	21372	238950	3260	6127
7	3	21302	283880	3037	5510
8	3	21199	328208	2852	5003
9	3	21126	373225	2693	4563
10	3			2701	4448
11	3			2616	4193
12	3			2538	3965
13	3			2462	3743
14	3			2448	3623
15	3			2366	3434
16	3			2335	3318
17	3			2306	3220
18	3			2270	3093
19	3			2240	2990
20	3			2235	2946
21	3			2165	2326
22	3			2111	2242
23	3			2116	2233
24	3			2067	2127

Table C.24: LongCipher, 5972, 15360, full length

N	t	E	D	E-CRT3	D-CRT3
1	3	91200	109582		
2	3	88052	103837	89886	57533
3	3	87918	144956	87822	52209
4	3	87959	189080	86980	49775
5	3	87754	233668	86293	48311
6	3	86628	276452	85216	46953
7	3	87725	324925	85692	46807
8	3	87396	369516	85256	46199
9	3	85817	408025	83589	44956
10	3			87853	47038
11	3			87842	46841
12	3			87602	46523
13	3			87240	46172
14	3			87181	46033
15	3			86705	45635
16	3			85044	44651
17	3			83442	43703
18	3			83188	43494
19	3			82938	43284
20	3			83166	43337
21	3			91504	46987
22	3			91292	46855
23	3			91328	46847
24	3			91280	46764

Appendix D

CL Runtimes - ShortCipher

In this appendix, we present the results of our implementation of the ShortCipher scheme. The table captions are same as the ones described in Appendix C, but the CL scheme is now ShortCipher. The encryption and decryption labels in table headers of this chapter now describe the ShortCipher variants of the ones mentioned in Appendix C, page 94, with E referring to Algorithm 3.4.2 and D referring to Algorithm 3.4.3.

Table D.1: ShortCipher, 1828, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	11	7					
1	2	11	7					
2	1	11	7	7	12	7	12	7

Table D.2: ShortCipher, 1828, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	40	20					
1	2	38	20					
2	1	39	19	19	39	19	39	19

Table D.3: ShortCipher, 1828, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	13	8					
1	2	13	9					
1	3	14	9					
1	4	14	9					
1	5	14	9					
2	1	13	9	9	13	9	13	9
2	2	14	10				13	9
2	3	14	10				13	9
2	4	14	10				14	10
2	5	14	10				14	10
3	1	13	9	9	13	9	13	9
3	2	14	10				14	10
4	1	13	9	9	13	9	13	9
4	2	14	11				14	10
5	1	13	9	9	13	9	14	9

Table D.4: ShortCipher, 1828, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	43	22					
1	2	42	23					
1	3	45	23					
1	4	44	23					
1	5	44	23					
2	1	44	22	22	44	22	44	22
2	2	42	23				42	23
2	3	45	23				44	23
2	4	44	23				43	23
2	5	45	24				44	23
3	1	43	22	22	43	22	43	22
3	2	44	23				44	23
4	1	42	22	22	43	22	42	22
4	2	45	24				44	23
5	1	44	22	22	44	22	44	22

Table D.5: ShortCipher, 1828, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	18	13					
1	2	20	15					
1	3	20	16					
1	4	20	17					
1	5	20	18					
2	1	18	13	13	18	13	18	13
2	2	20	17				19	15
2	3	20	18				19	16
2	4	20	18				19	16
2	5	20	19				19	17
3	1	18	14	14	18	14	18	14
3	2	20	19				19	15
3	3	20	20				19	16
3	4	20	20				19	16
3	5	20	21				19	17
4	1	18	13	13	18	13	18	13
4	2	20	21				19	15
4	3	20	22				19	16
4	4	20	21				19	16
4	5	20	22				19	16
5	1	18	13	14	18	14	18	14
5	2	20	22				19	15
5	3	20	24				19	16
5	4	20	23				18	16
5	5	20	24				19	16

Table D.6: ShortCipher, 1828, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	54	29					
1	2	56	31					
1	3	56	32					
1	4	56	33					
1	5	55	34					
2	1	53	29	29	53	29	53	29
2	2	56	33				55	31
2	3	56	34				55	32
2	4	56	34				55	32
2	5	56	35				55	33
3	1	55	30	30	55	30	55	30
3	2	56	35				55	31
3	3	56	35				55	32
3	4	56	36				55	32
3	5	56	37				55	33
4	1	54	29	29	54	29	54	29
4	2	56	37				55	31
4	3	56	38				55	32
4	4	56	37				55	32
4	5	56	38				54	32
5	1	55	30	30	55	30	55	30
5	2	56	38				54	31
5	3	57	40				55	32
5	4	55	39				54	32
5	5	55	40				54	32

Table D.7: ShortCipher, 1828, 3072, short length

N	t	E	D	E-CRT3	D-CRT3
1	2	596	1010		
2	2	592	1596	624	633
3	2	596	2549	593	602
4	2	596	3502	577	586
5	2	595	4477	566	574
6	2	598	5451	564	571
7	2	593	6422	555	562
8	2	590	7366	549	557
9	2	595	8347	551	559
10	2			550	557
11	2			550	558
12	2			550	557
13	2			548	556
14	2			548	556
15	2			546	553
16	2			547	554
17	2			543	551
18	2			546	554
19	2			548	556
20	2			545	552
21	2			546	552
22	2			546	553
23	2			545	552
24	2			546	553

Table D.8: ShortCipher, 1828, 3072, full length

N	t	E	D	E-CRT3	D-CRT3
1	2	746	1079		
2	2	739	1667	771	701
3	2	747	2619	744	669
4	2	743	3576	724	653
5	2	746	4550	716	641
6	2	749	5533	715	639
7	2	744	6505	706	631
8	2	740	7467	699	626
9	2	748	8461	703	628
10	2			703	627
11	2			701	625
12	2			698	624
13	2			698	623
14	2			698	623
15	2			695	619
16	2			694	620
17	2			685	617
18	2			694	621
19	2			708	632
20	2			699	627
21	2			706	629
22	2			700	623
23	2			699	622
24	2			691	623

Table D.9: ShortCipher, 3598, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	38	21					
1	2	39	21					
2	1	39	21	21	39	21	39	21

Table D.10: ShortCipher, 3598, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	173	79					
1	2	174	79					
2	1	175	79	79	175	79	175	79

Table D.11: ShortCipher, 3598, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	41	24					
1	2	44	24					
1	3	42	25					
1	4	43	25					
1	5	44	25					
2	1	42	24	24	42	24	42	24
2	2	44	25				43	25
2	3	44	25				42	25
2	4	44	25				42	25
2	5	44	26				43	25
3	1	41	24	23	42	24	42	24
3	2	43	26				42	25
4	1	42	24	24	43	24	43	24
4	2	44	26				43	25
5	1	42	24	24	43	24	43	24

Table D.12: ShortCipher, 3598, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	178	83					
1	2	182	84					
1	3	173	83					
1	4	179	83					
1	5	181	83					
2	1	180	83	83	181	83	180	83
2	2	181	84				180	83
2	3	179	83				177	83
2	4	179	83				177	83
2	5	181	84				180	84
3	1	176	82	82	176	82	176	82
3	2	178	84				176	83
4	1	179	82	82	179	82	179	82
4	2	181	85				180	84
5	1	176	81	81	177	81	177	81

Table D.13: ShortCipher, 3598, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	51	33					
1	2	56	35					
1	3	57	37					
1	4	56	37					
1	5	55	39					
2	1	52	33	33	52	33	52	33
2	2	56	37				52	35
2	3	55	38				51	36
2	4	57	39				53	36
2	5	56	40				52	37
3	1	51	33	33	52	33	52	33
3	2	55	39				51	35
3	3	56	40				52	36
3	4	56	41				52	37
3	5	57	42				53	38
4	1	52	33	33	52	33	52	33
4	2	55	41				51	35
4	3	56	43				52	36
4	4	56	43				53	36
4	5	56	43				52	37
5	1	51	33	33	52	33	52	33
5	2	57	44				53	36
5	3	57	46				53	36
5	4	56	45				53	36
5	5	57	46				53	37

Table D.14: ShortCipher, 3598, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	205	98					
1	2	210	100					
1	3	209	101					
1	4	206	102					
1	5	205	104					
2	1	206	99	99	206	99	206	99
2	2	210	103				206	100
2	3	205	103				202	101
2	4	210	104				207	102
2	5	210	105				207	103
3	1	204	99	98	204	99	204	99
3	2	202	105				199	100
3	3	207	105				203	100
3	4	207	106				204	101
3	5	208	107				204	102
4	1	205	98	98	205	98	205	98
4	2	204	106				200	100
4	3	207	108				204	101
4	4	209	108				206	101
4	5	207	107				204	101
5	1	205	98	98	205	98	205	98
5	2	209	109				206	101
5	3	210	110				206	101
5	4	210	109				206	101
5	5	210	111				207	102

Table D.15: ShortCipher, 3598, 7680, short length

N	t	E	D	E-CRT3	D-CRT3
1	3	7213	19430		
2	3	7126	21886	7835	9399
3	3	7182	33605	7401	8471
4	3	7218	45759	7212	8030
5	3	7196	58031	7039	7696
6	3	7224	70736	6998	7589
7	3	7193	83204	6899	7420
8	3	7198	95600	6871	7321
9	3	7190	108156	6809	7213
10	3			6778	7144
11	3			6738	7074
12	3			6766	7092
13	3			6746	7046
14	3			6715	7002
15	3			6704	6968
16	3			6721	6970
17	3			6687	6910
18	3			6694	6915
19	3			6651	6870
20	3			6566	6774
21	3			6577	6783
22	3			6674	6850
23	3			6700	6877
24	3			6695	6868

Table D.16: ShortCipher, 3598, 7680, full length

N	t	E	D	E-CRT3	D-CRT3
1	3	8126	19924		
2	3	8077	22535	8774	9878
3	3	8066	34146	8277	8881
4	3	8100	46338	8080	8427
5	3	8105	58817	7939	8123
6	3	8101	71328	7860	7962
7	3	8068	83948	7771	7812
8	3	8048	96286	7713	7710
9	3	8073	108990	7684	7610
10	3			7666	7554
11	3			7592	7468
12	3			7600	7442
13	3			7567	7381
14	3			7565	7369
15	3			7547	7329
16	3			7570	7338
17	3			7633	7376
18	3			7534	7277
19	3			7488	7230
20	3			7498	7253
21	3			7506	7289
22	3			7502	7245
23	3			7509	7258
24	3			7512	7285

Table D.17: ShortCipher, 5972, 16, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	95	49					
1	2	98	49					
2	1	97	49	49	98	49	98	49

Table D.18: ShortCipher, 5972, 16, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	540	240					
1	2	552	240					
2	1	549	240	239	549	240	550	240

Table D.19: ShortCipher, 5972, 80, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	105	55					
1	2	107	55					
1	3	108	56					
1	4	107	56					
1	5	108	56					
2	1	104	54	54	105	54	105	55
2	2	108	56				105	56
2	3	108	57				105	56
2	4	108	56				105	56
2	5	107	58				105	57
3	1	104	55	54	105	55	105	55
3	2	108	57				105	56
4	1	103	55	55	104	55	104	55
4	2	108	58				106	57
5	1	105	55	55	107	55	107	55

Table D.20: ShortCipher, 5972, 80, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	578	252					
1	2	576	252					
1	3	581	253					
1	4	577	253					
1	5	580	253					
2	1	579	252	252	579	252	579	252
2	2	582	254				579	254
2	3	582	254				580	254
2	4	582	254				579	254
2	5	573	255				571	254
3	1	578	252	252	579	252	579	252
3	2	578	254				576	253
4	1	564	251	251	565	251	566	251
4	2	582	255				580	254
5	1	578	252	252	579	252	579	252

Table D.21: ShortCipher, 5972, 256, short length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	121	71					
1	2	130	73					
1	3	129	75					
1	4	129	75					
1	5	129	77					
2	1	121	71	71	121	71	121	71
2	2	130	75				122	73
2	3	130	77				121	74
2	4	130	77				121	75
2	5	130	78				122	76
3	1	121	71	71	122	71	122	71
3	2	131	79				122	73
3	3	129	79				121	74
3	4	131	81				122	75
3	5	131	83				123	77
4	1	120	70	70	121	71	121	71
4	2	129	80				120	73
4	3	131	83				123	75
4	4	130	82				122	74
4	5	131	83				122	75
5	1	121	71	71	122	71	122	71
5	2	131	84				123	74
5	3	132	87				124	75
5	4	130	84				122	74
5	5	128	86				120	75

Table D.22: ShortCipher, 5972, 256, full length

N	t	E	D	D-CRT1	E-CRT2	D-CRT2	E-CRT3	D-CRT3
1	1	601	270					
1	2	611	274					
1	3	607	276					
1	4	606	276					
1	5	608	279					
2	1	606	273	273	606	273	607	273
2	2	612	276				605	273
2	3	612	277				605	274
2	4	609	278				602	275
2	5	612	279				606	276
3	1	606	273	273	607	273	608	273
3	2	615	280				608	275
3	3	607	280				600	275
3	4	611	282				604	276
3	5	615	285				608	279
4	1	605	274	274	607	274	608	274
4	2	609	283				603	276
4	3	627	291				620	283
4	4	629	291				621	284
4	5	631	292				624	284
5	1	621	280	280	622	280	623	280
5	2	631	293				624	283
5	3	631	295				624	284
5	4	632	294				625	284
5	5	615	295				608	283

Table D.23: ShortCipher, 5972, 15360, short length

N	t	E	D	E-CRT3	D-CRT3
1	3	26980	70982		
2	3	27091	81060	28927	34811
3	3	27001	123366	26925	30794
4	3	27134	168082	26154	29064
5	3	27027	213290	25574	27957
6	3	27016	258834	25234	27287
7	3	26957	303860	25020	26813
8	3	26921	349881	24764	26392
9	3	26911	394842	24633	26083
10	3			24648	25998
11	3			24488	25741
12	3			24382	25509
13	3			24752	25796
14	3			24379	25367
15	3			24325	25240
16	3			24344	25198
17	3			24276	25097
18	3			24225	24968
19	3			24098	24797
20	3			24227	24922
21	3			24035	24674
22	3			24114	24712
23	3			24127	24718
24	3			24159	24685

Table D.24: ShortCipher, 5972, 15360, full length

N	t	E	D	E-CRT3	D-CRT3
1	3	30075	72286		
2	3	30258	82299	32126	36135
3	3	30142	124556	30088	32085
4	3	30279	169226	29325	30369
5	3	30149	214288	28726	29257
6	3	30169	259829	28398	28569
7	3	30102	305174	28166	28141
8	3	30043	350888	27913	27670
9	3	30035	396305	27772	27397
10	3			27867	27335
11	3			27688	27090
12	3			27529	26837
13	3			27557	26717
14	3			27583	26726
15	3			27351	26459
16	3			27477	26476
17	3			27442	26403
18	3			27448	26341
19	3			27184	26114
20	3			27356	26145
21	3			27155	25931
22	3			27360	26184
23	3			27219	26046
24	3			27291	26064