# Notes on Lattice-Based Cryptography

Alessandro Budroni

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2022

UNIVERSITY OF BERGEN

# Notes on Lattice-Based Cryptography

Alessandro Budroni


Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 06.09.2022

Year: 2022

Title: Notes on Lattice-Based Cryptography

Name: Alessandro Budroni

Print: Skipnes Kommunikasjon / University of Bergen

# Acknowledgements

I take this opportunity to thank everyone who helped me in pursuing my Ph.D. and writing this thesis.

First of all, my gratitude goes to my doctoral supervisor, professor Igor Semaev, for his guidance and patience, for teaching me the right approach to scientific research, and for the many helpful and illuminating discussions. I want to thank my two co-supervisors, professor Chunlei Li, for his supportive advice, and professor Qian Guo, for introducing me to the research topic of the BKW algorithm for LWE and for our many research conversations.

I immensely enjoyed working with my coauthors and friends Andrea Tenti and Erik Mårtesson, and I am grateful to have met them during this journey. My other coauthor, Federico Pintore, was among the first to encourage me to pursue a Ph.D., and I am really grateful to him. Finally, I thank my other coauthors Benjamin Chetioui, Ermes Franch, Thomas Johansson, and Paul Stankowski, for the many discussions and the fruitful scientific collaborations.

These four years would not have been the same without the conducive work environment of the Selmer Center. I express my gratitude to all members of the group for the support and the nice time spent together. In particular, I want to thank Navid Ghaedi for our sincere friendship built on three years of laughs and office sharing. I am also very grateful for meeting many precious friends that supported me and allowed me to have the best time in Bergen. I cannot list them all for a matter of space, but I am extremely grateful to all my hiking, climbing, skiing, cooking, and juggling buddies.

I want to express my gratitude to my parents Giampiero and Stefania, for always being an example to follow, to my brothers Dario and Stefano, my grandma, and all the relatives who have supported me and encouraged me during these four years.

Last but not least, I thank my girlfriend and lifemate Malin for her love, support, and patient during the most challenging moments. To my little friend Emmi goes my gratitude for her unconditional love and for making me start every single day with a good mood.

# Abstract in English

Public-key Cryptography relies on the assumption that some computational problems are hard to solve. In 1994, Peter Shor showed that the two most used computational problems, namely the Discrete Logarithm Problem and the Integer Factoring Problem, are not hard to solve anymore when using a quantum computer. Since then, researchers have worked on finding new computational problems that are resistant to quantum attacks to replace these two. Lattice-based Cryptography is the research field that employs cryptographic primitives involving hard problems defined on lattices, such as the Shortest Vector Problem and the Closest Vector Problem. The NTRU cryptosystem, published in 1998, was one of the first to be introduced in this field. The Learning With Error (LWE) problem was introduced in 2005 by Regev, and it is now considered one of the most promising computational problems to be employed on a large scale in the near future. Studying its hardness and finding new and faster algorithms that solve it became a leading research topic in Cryptology. This thesis includes the following contributions to the field:

- A non-trivial reduction of the Mersenne Low Hamming Combination Search Problem, the underlying problem of an NTRU-like cryptosystem, to Integer Linear Programming (ILP). In particular, we find a family of weak keys.
- A concrete security analysis of the Integer-RLWE, a hard computational problem variant of LWE introduced by Gu Chunsheng. We formalize a meet-in-the-middle attack and a lattice-based attack for this case, and we exploit a weakness of the parameters choice given by Gu to build an improved lattice-based attack.
- An improvement of the Blum-Kalai-Wasserman algorithm to solve LWE. In particular, we introduce a new reduction step and a new guessing procedure to the algorithm. These allowed us to develop two implementations of the algorithm that are able to solve relatively large LWE instances. While the first one efficiently uses only RAM memory and is fully parallelizable, the second one exploits a combination of RAM and disk storage to overcome the memory limitations given by the RAM.
- We fill a gap in Pairing-based Cryptography by providing concrete formulas to compute hash-maps to $\mathbb{G}_2$, the second group in the pairing domain, for the Barreto-Lynn-Scott family of pairing-friendly elliptic curves.

# Abstract in Norwegian

Asymmetrisk kryptering er avhengig av antakelsen om at noen beregningsproblemer er vanskelige å løse. I 1994 viste Peter Shor at de to mest brukte beregningsproblemene, nemlig det diskrete logaritmeproblemet og primtallsfaktorisering, ikke lenger er vanskelige å løse når man bruker en kvantedatamaskin. Siden den gang har forskere jobbet med å finne nye beregningsproblemer som er motstandsdyktige mot kvanteangrep for å erstatte disse to. Gitterbasert kryptografi er forskningsfeltet som bruker kryptografiske primitiver som involverer vanskelige problemer definert på gitter, for eksempel det korteste vektorproblemet og det nærmeste vektorproblemet. NTRU-kryptosystemet, publisert i 1998, var et av de første som ble introdusert på dette feltet. Problemet Learning With Error (LWE) ble introdusert i 2005 av Regev, og det regnes nå som et av de mest lovende beregningsproblemene som snart tas i bruk i stor skala. Å studere vanskelighetsgraden og å finne nye og raskere algoritmer som løser den, ble et ledende forskningstema innen kryptografi. Denne oppgaven inkluderer følgende bidrag til feltet:

- En ikke-triviell reduksjon av Mersenne Low Hamming Combination Search Problem, det underliggende problemet med et NTRU-lignende kryptosystem, til Integer Linear Programming (ILP). Særlig finner vi en familie av svake nøkler.
- En konkret sikkerhetsanalyse av Integer-RLWE, en vanskelig beregningsproblemvariant av LWE, introdusert av Gu Chunsheng. Vi formaliserer et *meet-in-the-middle* og et gitterbasert angrep for denne saken, og vi utnytter en svakhet ved parametervalget gitt av Gu, for å bygge et forbedret gitterbasert angrep.
- En forbedring av Blum-Kalai-Wasserman-algoritmen for å løse LWE. Mer spesifikt, introduserer vi et nytt reduksjonstrinn og en ny gjetteprosedyre til algoritmen. Disse tillot oss å utvikle to implementeringer av algoritmen, som er i stand til å løse relativt store LWE-forekomster. Mens den første effektivt bare bruker RAM-minne og er fullt parallelliserbar, utnytter den andre en kombinasjon av RAM og disklagring for å overvinne minnebegrensningene gitt av RAM.
- Vi fyller et tomrom i paringsbasert kryptografi. Dette ved å gi konkrete formler for å beregne hash-funksjon til $\mathbb{G}_2$, den andre gruppen i paringsdomenet, for Barreto-Lynn-Scott-familien av paringsvennlige elliptiske kurver.

# List of Publications

In all the following papers, the authors are listed in alphabetical order.

I A. Budroni and A. Tenti, "The Mersenne Low Hamming Combination Search Problem Can Be Reduced to an ILP Problem," *Progress in Cryptology – AFRICACRYPT 2019*. Vol. 11627 of *LNCS*, pp. 41–55, Springer, 2019, Rabat, Morocco.

II A. Budroni, B. Chetioui and E. Franch, "Attacks on Integer-RLWE," *Information and Communications Security: 22nd International Conference (ICICS), Proceedings*. Vol. 12282 of *LNCS*, pp. 528–542, Springer, 2020, Copenhagen, Denmark.

III A. Budroni, Q. Guo, T. Johansson, E. Mårtensson and P. Stankovski Wagner, "Improvements on Making BKW Practical for LWE", Special Issue *Public-Key Cryptography in the Post-quantum Era* of *Cryptography*, MDPI, Vol 5, 2021.

This paper is an extended and improved version of the manuscript titled "Making the BKW Algorithm Practical for LWE", A. Budroni, Q. Guo, T. Johansson, E. Mårtensson and P. Stankovski Wagner, presented at the *21st International Conference on Cryptology* in India (*INDOCRYPT 2020*), Bangalore, India, and published in the proceedings.

IV A. Budroni and F. Pintore, "Efficient Hash Maps to $\mathbb{G}_2$ on BLS Curves", *Applicable Algebra in Engineering, Communication and Computing (AAECC)*, Springer, 2020.

This thesis includes revised versions of paper I, II and IV.

The development of the content of paper IV started when the author of this thesis was an employee at MIRACL Ltd. An early version of this work was presented as a poster titled "Hashing to $\mathbb{G}_2$ on BLS pairing-friendly curves" at the *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, 2018, New York, U.S., and an abstract was published in the proceedings.

# Contents

**Acknowledgements** i

**Abstract in English** iii

**Abstract in Norwegian** v

**List of Publications** vii

**I Introduction** 1

1 Introduction to Cryptology . . . 3

1.1 Symmetric Cryptography . . . 4

1.2 Asymmetric Cryptography . . . 5

1.3 Post-Quantum Cryptography . . . 8

1.4 Typical Attacks . . . 9

2 Preliminaries . . . 10

2.1 Complexity Estimation . . . 10

2.2 Lattices and Geometry of Numbers . . . 11

2.3 Distributions . . . 16

3 Foundations in Lattice-Based Cryptography . . . 19

3.1 NTRU . . . 19

3.2 Learning With Errors . . . 21

3.3 On Using the Ring of Integers $\mathbb{Z}$ . . . 24

4 Attacks to LWE . . . 27

4.1 Algebraic Attacks . . . 27

4.2 Combinatorial Attacks . . . . . 28
4.3 Lattice-based Attacks . . . . . 29
5 Pairing-based Cryptography . . . . . 31
5.1 The Group of Points of an Elliptic Curve . . . . . 31
5.2 Pairings in Cryptography . . . . . 32
5.3 Hash-Maps to $\mathbb{G}_1$ and $\mathbb{G}_2$ . . . . . 34
6 Contributions . . . . . 35

**II Included Papers 45**

**Paper 1: The Mersenne Low Hamming Combination Search Problem can be reduced to an ILP Problem 48**

1 Introduction . . . . . 49
1.1 Our Contribution/Outline . . . . . 50
2 Preliminaries . . . . . 51
2.1 Previous Attacks . . . . . 52
2.2 The Beunardeau et al. attack on MLHCombSP . . . . . 53
2.3 Integer Linear Programming . . . . . 54
3 ILP Reduction . . . . . 55
3.1 Merging Bits . . . . . 57
4 A new family of weak keys . . . . . 60
5 Conclusions and Future Work . . . . . 63

**Paper 2: Attacks on Integer-RLWE 68**

1 Introduction . . . . . 69

1.1 Contribution . . . . . 70

2 Preliminaries and Notation . . . . . 71

2.1 Discrete Gaussian Distributions . . . . . 71

2.2 Lattices . . . . . 72

2.3 Integer Ring-Learning With Errors . . . . . 73

3 Standard Attacks . . . . . 73

3.1 Meet-in-the-Middle attack . . . . . 73

3.2 Lattice-Based Attack . . . . . 77

4 Improved Lattice-Based Attack for Weak Choices of $n$ . . . . . 80

4.1 Analysis and Success Condition . . . . . 82

5 Experiments . . . . . 84

6 Conclusion . . . . . 85

**Paper 3: Improvements on Making BKW Practical for Solving LWE 90**

1 Introduction . . . . . 91

1.1 Related Work . . . . . 92

1.2 Contributions . . . . . 93

1.3 Organization . . . . . 93

2 Background . . . . . 94

2.1 Notation . . . . . 94

2.2 The LWE and LPN Problems . . . . . 94

2.3 Discrete Gaussian Distributions . . . . . 95

3 A Review of BKW-style Algorithms . . . . . 96

3.1 The LWE Problem Reformulated . . . . . 96

3.2 Transforming the Secret Distribution . . . . . 96
3.3 Sample Amplification . . . . . 97
3.4 Iterating and Guessing . . . . . 97
3.5 Plain BKW . . . . . 98
3.6 Coded-BKW and LMS . . . . . 98
3.7 LF1, LF2, Unnatural Selection . . . . . 98
3.8 Coded-BKW with Sieving . . . . . 99
4 BKW-style Reduction Using Smooth-LMS . . . . . 99
4.1 A New BKW-style Step . . . . . 99
4.2 Smooth-Plain BKW . . . . . 102
4.3 How to Choose the Interval Sizes $C_i$ . . . . . 102
4.4 Unnatural Selection . . . . . 103
4.5 On Optimizing $C_l$ Values . . . . . 103
4.6 An Illustration of Smooth Reduction Steps . . . . . 103
5 A Binary Partial Guessing Approach . . . . . 104
5.1 From LWE to LPN . . . . . 104
5.2 Guessing $\mathbf{s}_0$ Using the FWHT . . . . . 106
5.3 Retrieving the Original Secret . . . . . 107
6 Analysis of the Algorithm and its Complexity . . . . . 108
6.1 The Algorithm . . . . . 109
6.2 The Complexity of Each Step . . . . . 109
6.3 The Data Complexity . . . . . 111
6.4 In Summary . . . . . 113
6.5 Numerical Estimation . . . . . 113

7 Implementations . . . . . 113

7.1 RAM-based Implementation . . . . . 114

7.2 File-based Implementation . . . . . 115

7.3 A Novel Idea for Fast Storage Writing . . . . . 117

7.4 Other Implementation Aspects . . . . . 119

8 Experimental Results . . . . . 121

9 Conclusions and Future Work . . . . . 124

**Paper 4: Efficient Hash Maps to $\mathbb{G}_2$ on BLS curves** **130**

1 Introduction . . . . . 131

1.1 Pairings in Cryptography . . . . . 131

1.2 Families of pairing-friendly elliptic curves . . . . . 132

1.3 Hashing to $\mathbb{G}_2$ . . . . . 133

1.4 Related Work . . . . . 134

1.5 Contributions and Outline . . . . . 134

2 Known methods for efficiently mapping into $\mathbb{G}_2$ . . . . . 135

2.1 Scott *et al.* method . . . . . 135

2.2 Fuentes *et al.* method . . . . . 136

2.3 BLS curves . . . . . 139

3 Scott *et al.* method on BLS curves . . . . . 140

3.1 BLS-12 . . . . . 140

3.2 BLS-24 . . . . . 141

3.3 BLS-30 . . . . . 141

3.4 BLS-42 . . . . . 142

3.5 BLS-48 . . . . . 142

4 Fuentes *et al.* method on BLS curves with $k = 12, 24, 30$ . . . . . 143

4.1 BLS-12 . . . . . 143

4.2 BLS-24 . . . . . 144

4.3 BLS-30 . . . . . 144

5 Faster hash maps for BLS curves with $k = 42, 48$ . . . . . 145

5.1 BLS-48 . . . . . 145

5.2 BLS-42 . . . . . 147

6 Comparisons and conclusions . . . . . 148

# Part I

# Introduction

# 1 Introduction to Cryptology

> The difference between a key and an encryption key is that if you break the former, you cannot open the door, but if you break the latter, you open the door.
>
> *A. B.*

Cryptology is the science concerning secure communications between parties against adversaries. It encompasses both *Cryptography* and *Cryptanalysis*.

Cryptography, from Ancient Greek *kryptós* ("hidden") and *gráphein* ("to write"), includes the tools and techniques for transforming information by means of a secret *key* into a form that is either impossible or infeasible to reverse for an adversary.
Cryptanalysis, from Ancient Greek *kryptós* and *analýein* ("to untie"), includes the methods for recovering or forging secured information without the knowledge of the key. Nowadays, the term "Cryptography" is often used in the place of "Cryptology".

In the past, the security of a message often relied on the secrecy of the algorithm used in encryption. However, this approach of providing "security through obscurity" turned out to be unreliable when using the same system for a long time. The second of six principles introduced by Auguste Kerckhoffs in 1998, states that a cryptosystem should be secure even if the adversary knows everything about it, except the key [1]. Claude Shannon later reformulated this concept as "assume the enemy knows your system" [2], and most cryptographers nowadays accept it.

### General Framework

Here, we introduce the framework that we will use to describe different scenarios in Cryptography. Two entities or individuals, generally known in the literature as *Alice* and *Bob*, communicate through encryption: Alice encrypts a message and sends it over a public channel, Bob receives the encrypted message and decrypts it to read its content. An adversary, generally called *Eve*, has access to the public channel and, therefore, to the encrypted message. The goal of Eve is to recover or forge the original content of the message.

More formally, the encryption can be seen as a function $E$ that takes as input a

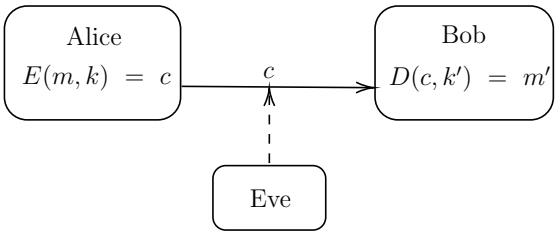


Figure 1: General framework in Cryptography.

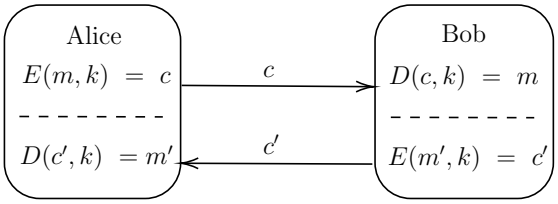


Figure 2: Symmetric encryption.

*plaintext* message $m$ and an encryption key $k$. The output of this function is known as the *ciphertext* $c = E(m,k)$. Similarly, the decryption process can be seen as a function $D$ that takes as input the ciphertext $c$ and a decryption key $k'$. The output is the message $m' = D(c,k')$. We say that the communication was successful when $m = m'$. The diagram in Figure 1 describes this general framework.

It is common to name research subfields of Cryptography according to their scope and the leading technologies used in their building blocks. For example, one can find Asymmetric and Symmetric Cryptography, Classical and Post-Quantum Cryptography, Elliptic-Curve Cryptography, Lattice-Based Cryptography, etc. In the following subsections, we introduce some of them to facilitate the understanding of this thesis. For a more detailed introduction to Cryptography and its history, we refer the reader to [3] and [4].

## 1.1 Symmetric Cryptography

Symmetric cryptosystems are characterized by having one single private key (a.k.a. symmetric-key or secret-key), generally known only by the two communicants Alice and Bob, used both in encryption and decryption. In other words, the encryption key $k$ and the decryption key $k'$ are the same ($k = k'$). Since both Alice to Bob are able to encrypt and decrypt, they have the same role in their bidirectional communication. This framework can be visualized in Figure 2.

### Hash functions

A hash function is a function $H$ that deterministically maps an array of data of any length $m$ to an array of a fixed length $h = H(m)$, also known as the *digest*. The properties that a hash function must satisfy to be considered cryptographically secure are the following:

1. **Pre-image resistance**: given $h = H(m)$, it is difficult to invert $H$ and obtain $m$.
2. **Second pre-image resistance**: given $m$, it is difficult to find $m' \neq m$ such that $H(m') = H(m)$.
3. **Collision resistance**: it is difficult to find two different $m$ and $m'$ such that $H(m') = H(m)$.

In particular, collision resistance implies pre-image and second pre-image resistance. Cryptographically secure hash functions are called *cryptographic hash functions* and are required to be efficient (fast) to compute.

## 1.2 Asymmetric Cryptography

A problem that arises when using symmetric cryptosystems is the exchange/agreement of the private key. Alice and Bob cannot securely communicate if they have not exchanged/agreed on a key before. Sending the key as cleartext is unsafe as a third party, Eve, can intercept it and use it to decrypt the following encrypted messages. This problem may be solved using asymmetric (or public-key) cryptosystems. This framework, employs two different keys: a *public key* used only in encryption (encryption key) and a *private key* used only in decryption (decryption key). Typically, even if they are related to each other, it is computationally infeasible to retrieve the private key from the public key. Anyone can encrypt messages using the public key, but only the possessor of the private key is able to decrypt them and read their content. The research field that studies asymmetric encryption is also known as *Public-key Cryptography*.

Over the past centuries, symmetric cryptosystems were the only ones known and used. In 1970, the British cryptographer James H. Ellis introduced the concept of public-key encryption [5]. However, he did not provide an instantiation of it. A few years later, in 1976, Whitfield Diffie and Martin Hellman published a work that introduced the key-agreement algorithm, nowadays known as the *Diffie-Hellman key exchange* (DH) [6]. In particular, it allows two parties to securely establish a shared encryption key by sharing random data and making some local computation. Even though DH does not allow to perform public-key encryption, it is generally considered a public-key cryptosystem.

In 1978, another public-key cryptosystem, generally known as RSA, was published by Ron Rivest, Adi Shamir, and Leonard Adleman [7]. Unlike DH, this allows to make public-key encryption, and it is based on the hard mathematical problem of factorizing large integers. Since then, many other public-key cryptosystems have been proposed.

More formally, a public-key cryptosystem works as follows. Alice generates a public key $k_{pb}$ and a private key $k_{pr}$. Then, she sends the public key over the public channel. Bob uses it to encrypt its message $c = E(m, k_{pb})$ and sends it to Alice, who decrypts it using her private key $m = D(c, k_{pr})$. See Figure 3 to visualize this framework in a diagram.

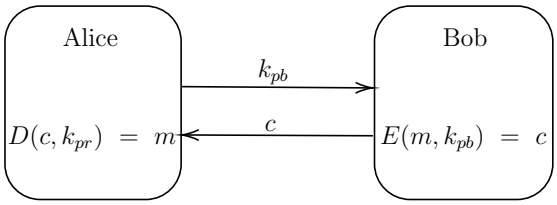


Figure 3: Asymmetric encryption.

In practice, asymmetric cryptosystems are usually used to exchange or agree on a secret that is then used as a *seed* (also known as *master secret*) from which a symmetric-key is generated. Then, a symmetric cryptosystem is employed to communicate efficiently. One of the most famous and used protocols that follows this approach is the *Transport Layer Security* (TLS) protocols [8].

### The RSA cryptosystem

To give the reader an example of public-key encryption, we give here a basic explanation of the RSA cryptosystem.

Let $p$ and $q$ be two large prime numbers and let $n = pq$. Let $\lambda(n)$ be an integer defined as $\text{lcm}(p-1, q-1)$[1], and let $e < \lambda(n)$ be a positive integer coprime to $\lambda(n)$. Let $d$ be the multiplicative inverse modulo $\lambda(n)$, i.e.

$$d = e^{-1} \mod \lambda(n).$$

The pair $(n, e)$ constitutes the public key, and $d$ is the private key. Let $0 \leq m < n$ be the plaintext, then Bob performs encryption and computes the ciphertext $c$ as follows

$$c = m^e \mod n.$$

[1]least common multiple: the smallest positive integer that is divisible by both $p-1$ and $q-1$.

Using the private key, Alice is able to decrypt the ciphertext and retrieve the message $m$ as follows

$$c^d = (m^e)^d = m^{ed} = m \mod n.$$

It is important to keep the parameters $\lambda(n)$, $p$ and $q$ must be kept secret[2]; otherwise, the cryptosystem is insecure. Knowing $\lambda(n)$ allows, indeed, to efficiently compute $d$ and, therefore, decrypt any message encrypted with the public key $(n, e)$.

The so-called *man-in-the-middle attack* is one of the basic approaches for attacking a public-key cryptosystem. An adversary, Eve, substitutes in the communication channel the public key of Alice $k_{pb}$ with another public key $k'_{pb}$, for which she owns the corresponding private key $k'_{pr}$. Then, Eve intercepts the encrypted message from Bob $c' = E(m, k'_{pb})$ and, therefore, is able to retrieve $m = D(c', k'_{pr})$. A countermeasure to this attack consists of *signing* the private key so that Bob can verify that $k_{pb}$ is actually the private key of Alice[3]. In the following section, we introduce a kind of cryptographic scheme that allows performing such a signature.

### Digital signature

A *digital signature* is a cryptographic scheme used to validate the authenticity and the integrity of messages sent over a channel. Usually, the sender "signs" the content of the message (e.g., email, money transaction, documents, etc.), allowing the receiver to verify its origin, state, and identity of the sender.

Let us consider the scenario of Alice sending a message $m$ to Bob. Assume Bob already posses Alice's public key $k_{pb}$. To sign the message, Alice applies as hash function on the message $h = H(m)$, and encrypts the digest using her private key, i.e. $s_m = E(h, k_{pr})$. Then, Bob decrypts it using Alice's public key $h' = D(s_m, k_{pb})$ (see Figure 4). By doing that, Bob can check whether $h$ is equal to $h'$. One distinguishes two scenarios:

- If $h = h'$, then Bob can conclude that $m$ was sent by Alice since, as the only possessor of the private key $k_{pr}$, she was the only one able to sign the document.
- If $h \neq h'$, then Bob cannot be sure whether the message received $m$ was actually sent by Alice or was not modified/substituted by a third party.

A few public-key cryptosystems provide both public-key encryption and digital signature. It is the case of the RSA cryptosystem. In this case, Alice signs the hash of her

[2]Knowing $p$ or $q$ allows to retrieve $\lambda(n)$.
[3]In practice, a third-party called Certificate Authority is involved in this process.

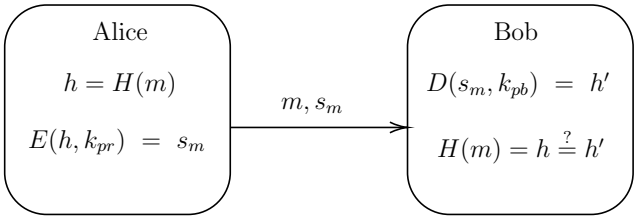


Figure 4: Digital signature.

message $h = H(m)$ with the private key $d$ as follows

$$s_m = h^d \mod n.$$

Using the public key $(n, e)$, Bob is able to retrieve the hash of the message as follows

$$s_m^e = (h^d)^e = h^{de} = h \mod n.$$

## 1.3 Post-Quantum Cryptography

Public-key cryptosystems base their security on hard mathematical problems. Namely, certain cryptographic schemes are proved to be secure as long as the complexity of the underlying mathematical problem is exponential (in some cases, sub-exponential). At the moment, the majority of public-key cryptographic schemes employed in real-life applications are based on either of the following two assumptions.

- **The Integer Factoring Assumption:** it is hard to find two big primes $p$ and $q$ given their product $N = pq$.
- **The Discrete Logarithm Assumption:** let $g$ be a generator for a cyclic group $G$. It is hard to find $x$, for a given element $h = g^x \in G$.

These are well-known and deeply-studied problems, and the best-known classical attacks to these have either exponential or sub-exponential complexity. In 1994, Peter Shor introduced a quantum algorithm that solves the Factoring Problem and the Discrete Logarithm Problem (DLP) in polynomial time [9]. Since then, the attention of the research community has moved to find new hard mathematical problems that resist quantum attacks too. In the last decade, due to the progress in constructing real quantum computers [10, 11], the research community received an additional boost to build quantum-resistant cryptographic schemes.

Most of the (believed to be) quantum-resistant cryptosystems proposed so far can be grouped into the following categories.

- **Lattice-based Cryptography**: the cryptographic primitives are based on the security of hard problems in high-dimensional lattices such as the Shortest Vector Problem or the Closest Vector Problem.
- **Code-based Cryptography**: the cryptographic primitives are based on the security of hard problems related to error correcting codes such as the Syndrome Decoding Problem.
- **Isogeny-based Cryptography**: the cryptographic primitives are based on the security of hard problems defined over supersingular elliptic curves and supersingular isogeny graphs.
- **Hash-based Cryptography**: the cryptographic primitives are based on the security of hash functions.
- **Multivariate Cryptography**: the cryptographic primitives are based on the hardness of solving systems of multivariate equations.

In 2017, the *National Institute of Standards and Technology* (NIST) launched a competition for standardizing new quantum-resistant public-key cryptosystems [12]. Initially, there were 82 submissions. At the moment of writing, 15 cryptosystems made it through three rounds of selection, and are divided into 7 *finalists*, considered the most promising ones, and 8 *alternates*. In particular, 5 out of 7 finalists and 2 out of 8 alternates are lattice-based cryptosystems.

## 1.4 Typical Attacks

There exist many cryptographic attacks to symmetric and asymmetric cryptosystems, and these can be categorized in different ways. Considering the goal of the adversary, we distinguish three cases:

- **Key-recovery attack:** the adversary's aim is to recover the private key.
- **Message-recovery attack:** the adversary's aim is to recover the encrypted message.
- **Distinguishing attack:** the adversary's aim is to distinguish encrypted messages from random data.

Another classification considers the information in hands of the adversary when performing the attack. The following are the main different scenarios that are usually taken into consideration:

- **Ciphertext-only attack (COA):** the adversary has access only to the ciphertext, and no information about the corresponding plaintext is known.
- **Known-plaintext attack (KPA):** the adversary has access to plaintext-ciphertext pairs.
- **Chosen-plaintext attack (CPA):** the adversary has the possibility to choose some specific plaintext and obtain the corresponding ciphertext.
- **Chosen-ciphertext attack (CCA):** the adversary has the possibility to choose some specific ciphertext and obtain the corresponding plaintext.

# 2 Preliminaries

Throughout this chapter, we use the following notation. Let $\mathbb{N}, \mathbb{Z}$ and $\mathbb{R}$ be the set of natural, integer, and real numbers respectively. Given a positive integer $p$, we write $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Vectors are denoted with bold lower case letters, matrices with bold upper case letters. For a given vector $\boldsymbol{v}$, we denote it's $i$-th component as $v_i$. We denote with $\| \boldsymbol{v} \|$ the Euclidean norm of $\boldsymbol{v}$, with entries $v_i \in \mathbb{R}$. For a good introduction to algebraic structures, we refer the reader to [13].

## 2.1 Complexity Estimation

It is fundamental to estimate the computational and memory resources necessary for solving a given algorithmic task in Cryptology.

One concrete approach consists of computing the number of unit operations that an algorithm needs to perform to complete a specific task. A unit operation can be an addition or a multiplication in a specific field, or even a bit-wise operation. Similarly, one can decide for a memory unit such as a bit, a byte, or an $n$-bits integer and compute the number of memory units required to execute the algorithm. However, it is sometimes difficult to predict the behavior of certain algorithms and, therefore, the number of operations.

Another approach is to use asymptotic analysis theory to study the behavior of an algorithm when the size of the problem goes to infinity. The so-called *Big O notation* is a set of notations commonly used in the literature, and it is briefly summarized in Table 2.1. It allows us to describe an algorithm with a bound on its asymptotic complexity. This strategy is particularly useful to classify algorithms and make comparisons. However, as we study the behavior *at infinity* of the algorithm, it may happen that the speed hierarchy is not respected for small problems. In other words, it may happen that, even if algorithm A has lower asymptotic complexity than algorithm B, A is slower than B for problems of (smaller) size used in real-world.

| Notation | Definition |
|---|---|
| $f(n) = \mathcal{O}(g(n))$ | $\limsup_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} < \infty$ |
| $f(n) = \Omega(g(n))$ | $\liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0$ |
| $f(n) = \Theta(g(n))$ | $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$ |
| $f(n) = o(g(n))$ | $\lim_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} = 0$ |
| $f(n) = \omega(g(n))$ | $\lim_{n\to\infty} \frac{\lvert f(n)\rvert}{g(n)} = \infty$ |
| $f(n) = \tilde{\mathcal{O}}(g(n))$ | $\exists k > 0 : f(n) = \mathcal{O}(g(n)\log^k g(n))$ |

Table 2.1: The Big O notation

One last approach is to run an implementation of the algorithm and measure the performance. This strategy allows us to have an exact measure of the time and memory needed to run the algorithm for a specific problem's size. The drawback is that the measurement would highly depend on the implementation and the machine. Also, it is often not straightforward to reach conclusions on the algorithm's behavior on large problem instances based on its performance with smaller ones.

## 2.2 Lattices and Geometry of Numbers

We recall here some basic definitions and notions useful to address the following sections on Lattice-Based Cryptography. For a more detailed introduction to the topic, we refer the reader to [14], [15] and [16].

**Definition 1.** (Lattice) *Let $k \leq n$ be two positive integers, and let $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k\} \subset \mathbb{R}^n$ a set of $k$ linearly independent vectors. The lattice $\Lambda$ of rank $k$ generated by $B$ is the set defined as follows:*

$$\Lambda = \Lambda(\mathcal{B}) = \{\alpha_1\boldsymbol{b}_1 + \alpha_2\boldsymbol{b}_2 + ... + \alpha_k\boldsymbol{b}_k,\ \alpha_i \in \mathbb{Z}\}\,.$$

*The set $\mathcal{B}$, or any other set of $k$ linearly independent vectors that generates $\Lambda$ as a $\mathbb{Z}$-module, is said to be a basis of $\Lambda$.*

More generally, a lattice is an additive sub-group of $\mathbb{R}^n$. Unless differently specified, from now on we will treat *full-rank* lattices, that is the case when $k = n$.

**Definition 2.** (Fundamental Domain) *Let $\mathcal{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_n\} \subset \mathbb{R}^n$ be a basis of an $n$-dimensional lattice $\Lambda$. The fundamental domain of $\Lambda$ corresponding to $\mathcal{B}$ is defined as the set*

$$\mathcal{F}(\mathcal{B}) = \{l_1\boldsymbol{b}_1 + l_2\boldsymbol{b}_2 + ... + l_n\boldsymbol{b}_n : \ 0 \leq l_i < 1\}.$$

**Definition 3.** (Volume of a Lattice) *Let $\Lambda$ be a lattice and let $\mathcal{F}$ be a fundamental domain of $\Lambda$. The volume (or determinant) of $\Lambda$ is defined as the volume of $\mathcal{F}$ and it is denoted by* $\mathrm{Vol}(\Lambda)$ *(or* $\det(\Lambda)$*).*

In particular, it can be shown that the volume of a lattice is invariant from the choice of the basis.

**Proposition 1.** *Let $\boldsymbol{B}$ be the $n \times n$ matrix whose rows are the vectors of a basis $\mathcal{B}$ of a lattice $\Lambda$. Then*

$$\mathrm{Vol}(\Lambda) = \mathrm{Vol}(\mathcal{F}) = |\det(\boldsymbol{B})|.$$

We denote with $\lambda_1(\Lambda)$ the length of the shortest non-zero vector in the lattice $\lambda$, i.e.,

$$\lambda_1(\Lambda) = \min_{\boldsymbol{v} \in \Lambda \setminus \{\boldsymbol{0}\}} \| \boldsymbol{v} \| .$$

This parameter plays a fundamental role in Lattice-based Cryptography as it is the subject of several computational problems defined over lattices. More in general, let $B_n(r)$ be the closed ball of radius $r$ in $\mathbb{R}^n$. Then, for $i = 1, ..., n$, define the $i^{th}$ *minima* of the lattice $\Lambda$ as

$$\lambda_i(\Lambda) = \min\{r \in \mathbb{R}^+ : \Lambda \cap B_n(r) \text{ contains } i \text{ linearly independent vectors}\}.$$

### Hermite's Theorem and Gaussian Heuristic

In Cryptography, it is often required to estimate the length of a shortest non-zero vector $\lambda_1(\Lambda)$ of a given lattice. In this section we report some well-known results on this topic. For a more detailed explanation, we refer the reader to [17].

**Definition 4.** (Hermite's Constant) *The supremum of $\lambda_1(\Lambda)^2 / \mathrm{Vol}(\Lambda)^{2/k}$, over all lattices $\Lambda \subset \mathbb{R}^n$ of rank $k$, is called Hermite's constant and it is denoted by $\gamma_k$.*

In other words, for a full-ranked lattice $\Lambda$, one can write

$$\lambda_1(\Lambda)^2 \leq \gamma_n \operatorname{Vol}(\Lambda)^{2/n}. \tag{2.1}$$

The exact value of $\gamma_n$ is known only for a few small values of $n$, in particular $\gamma_2 = \sqrt{4/3}$. The following theorem gives us a bound for $\gamma_n$.

**Theorem 1.** (Hermite's Theorem)[18] *For all $n \geq 2$, one has that $\gamma_n \leq \gamma_2^{n-1}$.*

Using Theorem 1 and relation (2.1), one gets the following bound on the length of the shortest vector of a lattice

$$\lambda_1(\Lambda) \leq \gamma_2^{(n-1)/2} \operatorname{Vol}(\Lambda)^{1/n} = (4/3)^{(n-1)/4} \operatorname{Vol}(\Lambda)^{1/n}.$$

Tighter bounds for $\gamma_n$ are given in [19, Chap. 9] and [20, Chap. 2], resulting in the following

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} + o(1) \leq \gamma_n \leq \frac{1.744n}{2\pi e}(1 + o(1)).$$

However, it is often more convenient to use heuristic estimates on random lattices to estimate the value of $\lambda(\Lambda)$. We report below an estimate based on the so-called *Gaussian Heuristic*.

**Heuristic 1.** (Gaussian Heuristic) *Let $\Lambda$ be a full-rank lattice and let $S \subset \mathbb{R}^n$ be a measurable set. Then, the number of points of $\Lambda \cap S$ is approximately* $\operatorname{Vol}(S)/\operatorname{Vol}(\Lambda)$.

Choose the set $S$ to be the $n$-dimensional ball $B_n(r)$, where the radius $r$ is such that $\operatorname{Vol}(B_n(r)) = \operatorname{Vol}(\Lambda)$. Then, according to Heuristic 1, $B_n(r)$ contains only one vector of $\Lambda$ (up to sign), and $\lambda_1(\Lambda)$ can be approximated with the radius $r$. Given that $\operatorname{Vol}(B_n(r))^{1/n}$ is approximately equal to $\sqrt{\frac{2\pi e}{n}}r$, one gets that

$$\lambda_1(\Lambda) \approx \sqrt{\frac{n}{2\pi e}} \cdot \operatorname{Vol}(\Lambda)^{1/n}. \tag{2.2}$$

Although this is a derivation of the Heuristic 1, it is often referred to in the literature as Gaussian Heuristic.

### Computational problems

Here, we define some of the computational problems defined on lattices that have proved to be useful in cryptographic applications. We start by introducing two fundamental problems.

**Definition 5.** (Shortest Vector Problem (SVP)) *Given a basis $\mathcal{B}$ of an $n-$dimensional lattice $\Lambda$, find a vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v}\| = \lambda_1(\Lambda)$.*

**Definition 6.** (Closest Vector Problem (CVP)) *Given a basis $\mathcal{B}$ of an $n-$dimensional lattice $\Lambda$ and a target vector $\boldsymbol{w} \in \mathbb{R}^n$, find a vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{w} - \boldsymbol{v}\| = \min_{\boldsymbol{t} \in \Lambda \setminus \{\boldsymbol{w}\}} \|\boldsymbol{w} - \boldsymbol{t}\|$.*

One can see the SVP as a particular case of the CVP for when the target vector is the zero vector. Particularly useful in Cryptography are the so-called *approximation* problems, i.e. parametrized versions of the above. Specifically, one usually uses a parameter $\gamma = \gamma(n)$ expressed as a function of the dimension $n$ of the lattice. The following is the approximate version of the SVP.

**Definition 7.** (Approximate Shortest Vector Problem ($\text{SVP}_\gamma$)) *Let $\gamma \geq 1$ be a real number. Given a basis $\mathcal{B}$ of an $n$-dimensional lattice $\Lambda$, find a non-zero vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(\Lambda)$.*

Note that, when $\gamma = 1$, then the above corresponds to the plain SVP. The following are two other important problems in lattice-based cryptography.

**Definition 8.** ($\gamma-$unique Shortest Vector Problem (unique-$\text{SVP}_\gamma$)) *Let $\gamma \geq 1$ be a real number. Given a lattice $\Lambda$ such that $\lambda_2(\Lambda) > \gamma\lambda_1(\Lambda)$, find the unique (up to sign) $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v}\| = \lambda_1(\Lambda)$.*

**Definition 9.** (Bounded Distance Decoding (BDD) *Let $d > 0$ be a real number. Given a basis $\mathcal{B}$ of an $n-$dimensional lattice $\Lambda$ and a target vector $\boldsymbol{w} \in \mathbb{R}^n$ such that there exists a unique lattice vector $\boldsymbol{v} \in \Lambda$ where $\|\boldsymbol{w} - \boldsymbol{v}\| < d$, find $\boldsymbol{v}$.*

### Lattice Reduction Algorithms

It is easy to obtain a basis of a lattice from another. Let $\boldsymbol{B} \in \mathbb{Z}^{n \times n}$ be a matrix whose rows generate a lattice $\Lambda$. For any matrix $\boldsymbol{U} \in \mathbb{Z}^{n \times n}$ with $\det(\boldsymbol{U}) = \pm 1$, the rows of $\boldsymbol{B}' = \boldsymbol{U}\boldsymbol{B}$ generate the same lattice $\Lambda$. A challenge in Lattice Theory is to find, given a basis of a lattice, another basis such that this is *reduced*, i.e., the vectors are (almost) orthogonal and have a small norm. In this section, we recall some of the most important *reduction algorithms* that address this problem.

**The LLL algorithm** In 1982, Lenstra, Lenstra, and Lovász introduced a polynomial-time algorithm (LLL) to reduce bases of lattices [21]. In this section, we report some basic notions and results on the complexity of the LLL algorithm and the *quality* of the basis it allows to compute.

**Definition 10.** (Gram-Schmidt Orthogonalization) *Let $\mathcal{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{R}^n\}$ be a set of linearly independent vectors in $\mathbb{R}^n$. The Gram-Schmidt Orthogonalization of $\mathcal{B}$ is the set of vectors denoted as $\mathcal{B}^* = \{\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^* \in \mathbb{R}^n\}$ and defined as follows:*

$$\boldsymbol{b}_1^* = \boldsymbol{b}_1, \qquad \boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \frac{\boldsymbol{b}_i \cdot \boldsymbol{b}_j^*}{\|\boldsymbol{b}_j^*\|^2} \boldsymbol{b}_j^*, \quad \text{for } 1 < i \leq n.$$

**Definition 11.** (LLL-reduced basis) *Let $\mathcal{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_n\}$ be a basis for an $n-$dimensional lattice $\Lambda$ and let $\mathcal{B}^* = \{\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*\}$ be its associated Gram-Schmidt Orthogonalization. The basis $\mathcal{B}$ is said to be LLL-reduced if it satisfies the following conditions:*

$$|\mu_{i,j}| = \frac{|\boldsymbol{b}_i \cdot \boldsymbol{b}_j^*|}{\|\boldsymbol{b}_j^*\|^2} \leq \frac{1}{2}, \qquad \text{for all } 1 \leq j < i \leq n, \qquad \text{(Size Condition)}$$

$$\|\boldsymbol{b}_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|\boldsymbol{b}_{i-1}^*\|^2, \qquad \text{for all } 1 < i \leq n. \qquad \text{(Lovász Condition)}$$

Theorem 2 gives us the complexity of LLL algorithm. We refer the reader to the original paper [21] for a detailed description of the algorithm.

**Theorem 2.** (LLL Algorithm) *The LLL algorithm, on a given input basis $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_n\}$ of an $n-$dimensional lattice $\Lambda$, terminates and returns an LLL-reduced basis $\{\boldsymbol{b}_1, ..., \boldsymbol{b}_n\}$ for $\Lambda$ in time*

$$\mathcal{O}(n^6 \log^3 B),$$

*where $B = \max_i \|\boldsymbol{v}_i\|$.*

An LLL-reduced basis $\{\boldsymbol{b}_1, ..., \boldsymbol{b}_n\}$ of an $n-$dimensional lattice $\Lambda$ comes with the following important property:

$$\|\boldsymbol{b}_1\| \leq 2^{(n-1)/2} \lambda_1(\Lambda).$$

Therefore, the LLL algorithm solves the $\text{SVP}_\gamma$ for $\gamma = 2^{(n-1)/2}$.

**The BKZ algorithm** In 1987, Schnorr introduced a generalization of the LLL algorithm that is nowadays known as the BKZ algorithm [22]. This was then further improved and instantiated in 1993 by Schnorr and Euchner [23]. The BKZ algorithm offers a trade-off between time complexity and quality of the reduced basis. In particular, an SVP oracle is used as a sub-routine to solve the SVP on a lattice of smaller dimension $\beta$. For $\beta = 2$, the BKZ algorithm corresponds to the LLL algorithm. When increasing $\beta$, the quality of the reduced basis improves at the cost of increasing the overall time complexity.

It has been shown that the BKZ algorithm makes a polynomial number of calls to the SVP oracle before it terminates [24]. Therefore, to estimate the overall complexity, one generally considers the complexity of this sub-routine. There are two main approaches for SVP oracle implementation: *lattice enumeration* and *lattice sieving*.

Lattice enumeration was the first one to be implemented. While it requires only a polynomial amount of space in the problem's size, it comes with super-exponential time complexity [25, 26]. On the other hand, sieving algorithms have an exponential complexity in time and memory that can be expressed as $2^{c\beta+o(\beta)}$, for a constant $c$. Recent asymptotic improvements [27, 28] fixed the constant $c$ to 0.292 for the case of sieving algorithms running on classical computers. A quantum speed-up allows lowering it down to $c = 0.265$ [29]. However, since these algorithms require building a list of lattice vectors of size $(4/3)^{\beta/2}$, one can consider $c = \log_2 \sqrt{4/3} = 0.2075$ for a conservative lower bound of their complexity. Also, a recent implementation of BKZ with lattice sieving overcame the results obtained with enumeration [30].

## 2.3 Distributions

In this section, we introduce some probability distributions that are useful for our scope.

**Definition 12.** (Probability Mass Function) *Let $X$ be a discrete random variable with support $R_X = \{x_1, x_2, ...\}$ finite or countably infinite. The function $p_X$ defined as*

$$p_X : R_X \to [0, 1], \qquad x \mapsto p_X(x) = Pr(X = x),$$

*is called the probability mass function (PMF) of $X$.*

The probability mass function defines the *probability distribution* of a random variable. We write $X \sim D$ to say that $X$ is with probability distribution $D$. In the literature, the corresponding of the PMF for the continuous case is often called *probability density function* (pdf).

**Definition 13.** (Uniform Distribution) *Let $X$ be a random variable ranging over a finite set $R$ with $n$ elements, and let its PMF be defined as*

$$Pr(X = x) = 1/n, \qquad \text{for every } x \in R.$$

*Then $X$ is said to be uniformly distributed over $R$, and we write $X \sim \mathcal{U}(R)$.*

**Definition 14.** (Normal Distribution) *A random variable $X$ is said to be normally distributed with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma > 0 \in \mathbb{R}$ if its probability density*

*function is defined as*

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \qquad \textit{for every } x \in \mathbb{R}$$

*and we write* $X \sim \mathcal{N}(\mu, \sigma)$.

**Definition 15.** (Discrete Normal Distribution) *A random variable* $X$*, with support* $\mathbb{Z}$*, is said to follow the Discrete Normal Distribution with mean* 0 *and variance* $\sigma^2$*, denoted with* $\mathcal{D}_{\sigma,\mathbb{Z}}$*, if its PMF is defined as*

$$\frac{e^{-x^2/2\sigma^2}}{\sum_{b\in\mathbb{Z}} e^{-b^2/2\sigma^2}}, \qquad \textit{for every } x \in \mathbb{Z}.$$

The Normal Distribution is often used to approximate its discrete counterpart. In particular, given $X \sim \mathcal{D}_{\sigma_1,\mathbb{Z}}$ and $Y \sim \mathcal{D}_{\sigma_2,\mathbb{Z}}$, one uses the properties valid for the continuous case and approximates the distribution of $X + Y$ with $\mathcal{D}_{\sqrt{\sigma_1^2+\sigma_2^2},\mathbb{Z}}$.

**Definition 16.** (Statistic) *Let* $X_1, ..., X_n$ *be random variables, and let* $T(x_1, ...x_n)$ *be a real-valued function whose domain includes the sample space of* $(X_1, ..., X_n)$*. Then, the random variable* $Y = T(X_1, ..., X_n)$ *is called statistic.*

An *hypothesis* is a statement about a parameter of a population. In an hypothesis testing problem, denote with $H_0$ and $H_1$ the two complementary hypothesis, called respectively *null* and *alternative hypothesis*.

**Definition 17.** (Hypothesis Test) *A hypothesis test is a rule that specifies:*

- *For which sample values the decision is made to accept* $H_0$ *as true.*
- *For which sample values* $H_0$ *is rejected and* $H_1$ *is accepted as true.*

For random independent samples $X_1, ...X_n$ from a population with PMF (of pdf) $f(x|\theta)$, for a parameter $\theta \in \Theta$, the *likelihood function* is defined as

$$L(\theta|\boldsymbol{x}) = L(\theta|x_1, ...x_n) = \prod_{i=1}^{n} f(x_i|\theta).$$

The likelihood ratio test statistic for testing $H_0 : \theta = \theta_0$ versus $H_1 : \theta = \theta_1$ is

$$\lambda(\boldsymbol{x}) = \frac{L(\theta_0|\boldsymbol{x})}{L(\theta_1|\boldsymbol{x})}.$$

A *likelihood ratio test* is any test that has rejection region of the form $R = \{\boldsymbol{x} : \lambda(\boldsymbol{x}) \leq c\}$, for some constant $c$, and level $\alpha = Pr(\boldsymbol{x} \in R|H_0)$.

The Neyman-Pearson Lemma [31] states that the likelihood ratio test is a *uniformly most powerful* $\alpha$ level test (see [32, Sec. 8.3.2]).

# 3 Foundations in Lattice-Based Cryptography

This section presents a survey on some of the computational problems in lattice-based cryptography that are of particular interest for this thesis. We refer the reader to [14], [15], and [16] for more detailed surveys on the topic.

## 3.1 NTRU

In 1998, Hoffstein, Pipher, and Silverman introduced NTRU, a new public-key cryptosystem that uses polynomial rings over a finite field as building blocks [33]. It includes two algorithms: *NTRUencrypt* and *NTRUsign* that are respectively for public-key encryption and digital signature. In this section, we follow the notation in [16, Chap. 7] to discuss the underlying problem of key-recovery in NTRUencrypt and its connection with lattices.

Let $n$ be a prime and let $p$ and $q$ be two moduli such that $\gcd(n, q) = \gcd(p, q) = 1$. Let $R$, $R_p$ and $R_q$ be polynomial rings defined as follows:

$$R = \mathbb{Z}[x] \Big/ (x^n - 1), \qquad R_p = \mathbb{Z}_p[x] \Big/ (x^n - 1), \qquad R_q = \mathbb{Z}_q[x] \Big/ (x^n - 1).$$

Given a polynomial $a(x) \in R$, we denote with $a_k$ its $k-$th coefficient, for $0 \leq k < n$. We use the notation $a(x) \star b(x)$ to write the product of two polynomials $a(x), b(x) \in R$. In particular, one has that

$$a(x) \star b(x) = c(x), \qquad \text{with} \quad c_k = \sum_{0 \leq i,j < n \ \mid \ i+j \equiv k \mod n} a_i b_j.$$

For two positive integers $d_1, d_2$, let $\mathcal{T}(d_1, d_2)$ be the set of polynomials in $R$ with exactly $d_1$ coefficients equal to 1, $d_2$ coefficients equal to $-1$, and all the others equal to 0.

### Key Generation, Encryption and Decryption

Let $N$, $p$, $q$, and $d$ be positive integers with $N$ and $p$ prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d + 1)p$. Choose at random $f(x) \in \mathcal{T}(d + 1, d)$ invertible in $R_p$ and $R_q$, and let $F_p(x)$ and $F_q(x)$ be its inverses in the two polynomial rings respectively. Choose at random $g(x) \in \mathcal{T}(d, d)$ and let it be the private key, and let $h(x) = F_q(x) \star g(x) \bmod q$ be the public key.

To encrypt a message $m(x) \in R_p$, one chooses at random $r \in \mathcal{T}(d, d)$, and computes the ciphertext as

$$c(x) = p \cdot r(x) \star h(x) + m(x) \mod q.$$

The decryption works as follows. Compute $f(x) \star c(x) = p \cdot g(x) \star r(x) + f(x) \star m(x)$ mod $q$. The result is then *center-lifted*[4] to $a(x) \in R$, and, finally the message is retrieved by computing

$$m(x) = F_p(x) \star a(x) \mod p.$$

The proof of the correctness of NTRUencrypt can be found in [16, Prop. 7.48].

### Key Recovery with Lattice Reduction

The following definition formalizes the underlying problem of key-recovering on NTRUencrypt.

**Definition 18.** (NTRU Key-Recovery Problem) *Let $d$ be a positive integer and let $f(x) \in \mathcal{T}(d+1, d)$ and $g(x) \in \mathcal{T}(d, d)$ be chosen at random such that $f(x)$ is invertible in $R_q$ and $R_p$. Let $h(x) \in R$ be such that*

$$f(x) \star h(x) \equiv g(x) \mod q. \tag{3.1}$$

*The NTRU Key-Recovery Problem consists of recovering $f(x)$ and $g(x)$ given $h(x)$.*

Note that, if $(f(x), g(x))$ satisfies (3.1), then $(x^k \star f(x), x^k \star g(x))$, for $0 \leq k < n$, is a solution of the problem too. Hence, the solution is not unique.

NTRU is usually classified as a lattice-based cryptosystem because the NTRU Key-Recovery Problem can be translated into a Shortest Vector Problem on a type of lattice called NTRU lattice. In this way, one can make an attack based on lattice reduction algorithms. Similarly, a plaintext-recovery attack can be translated into a Closest Vector Problem.

**Definition 19.** (NTRU lattice) *Let $\boldsymbol{h} = (h_0, h_1, ..., h_{n-1})$ be the vector of coefficients of an NTRU public key $h(x)$. Define the NTRU lattice $\Lambda_{\boldsymbol{h}}^{NTRU}$ associated with $h(x)$ as the*

[4]The center-lift of a polynomial $a'(x) \in R_q$ to $R$ is the unique polynomial $a(x) \in R$ satisfying $a(x)$ mod $q = a'(x)$, and with coefficients between $-q/2$ and $q/2$.

*lattice generated by the rows of the* $2n \times 2n$ *matrix*

$$\boldsymbol{M}_{\boldsymbol{h}}^{NTRU} = \left(\begin{array}{c|c} \boldsymbol{I}_n & \boldsymbol{H} \\ \hline \boldsymbol{O} & q\boldsymbol{I}_n \end{array}\right) = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right) \in \mathbb{Z}^{2n \times 2n}.$$

Let $f(x) \in \mathcal{T}(d+1, d)$ and $g(x) \in \mathcal{T}(d, d)$ be such that $f(x) \star h(x) \equiv g(x) \mod q$. Then, one has that

$$f(x) \star h(x) = g(x) + q \cdot u(x),$$

for some $u(x) \in R$. Let $\boldsymbol{f}, \boldsymbol{g}$ and $\boldsymbol{u}$ be the coefficient vectors of $f(x), g(x)$, and $u(x)$ respectively. It is easy to show that

$$(\boldsymbol{f}, -\boldsymbol{u})\boldsymbol{M}_{\boldsymbol{h}}^{\text{NTRU}} = (\boldsymbol{f}, \boldsymbol{f} \cdot \boldsymbol{H} - q \cdot \boldsymbol{u}) = (\boldsymbol{f}, \boldsymbol{g}).$$

Therefore, $(\boldsymbol{f}, \boldsymbol{g})$ can be obtained as a linear combination of the rows of $\boldsymbol{M}_{\boldsymbol{h}}^{\text{NTRU}}$ and, by definition, it is a vector of the lattice $\Lambda_{\boldsymbol{h}}^{\text{NTRU}}$. For the parameter's choices used in practice, $(\boldsymbol{f}, \boldsymbol{g})$ is a shortest vector of $\Lambda_{\boldsymbol{h}}^{\text{NTRU}}$. In this case, solving the SVP for the NTRU lattice $\Lambda_{\boldsymbol{h}}^{\text{NTRU}}$ is equivalent to finding $f(x)$ and $g(x)$. In practice, one applies a lattice reduction algorithm on $\boldsymbol{M}_{\boldsymbol{h}}^{\text{NTRU}}$ to retrieve the private key.

The above and its variants are considered the most efficient approaches to build a key-recovery attack for NTRU. For a more detailed analysis of the attacks to NTRU, we refer the reader to [34].

## 3.2 Learning With Errors

Introduced by Regev in 2005 [35], the Learning With Errors (LWE) problem is nowadays considered the most critical computational hard problem in Lattice-based Cryptography. It comes with reductions from worst-case lattice problems to average-case LWE, solidifying the confidence of its hardness [35, 36]. One can use it to build several cryptographic constructions with different features such as Key-Encapsulation Methods, Public-Key Encryption, Fully Homomorphic Encryption, etc.

**Definition 20.** (Learning With Errors Problem) *Let* $m \geq n$ *be positive integers and let*

*$\alpha, c_q > 0$ be two small parameters. Let $q = \mathcal{O}(n^{c_q})$ be prime and let $\chi_\sigma$ be a non-uniform probability distribution on $\mathbb{Z}$ with mean 0 and standard deviation $\sigma = \alpha q$. Given a secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ chosen uniformly at random, denote with $L_{\boldsymbol{s},\chi_\sigma}$ the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random and $e$ from $\chi_\sigma$, and returning*

$$(\boldsymbol{a},\ b = \boldsymbol{a} \cdot \boldsymbol{s} + e \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q. \tag{3.2}$$

- *The Search version of the Learning With Errors (SLWE) problem consists of, given $m$ samples from $L_{\boldsymbol{s},\chi_\sigma}$, recovering the secret vector $\boldsymbol{s}$.*
- *The Decision version of the Learning With Errors (DLWE) problem consists of, given $m$ pairs $(\boldsymbol{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, deciding whether these are sampled from $L_{\boldsymbol{s},\chi_\sigma}$ or uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Unless differently specified, we will assume that $\chi_\sigma$ is chosen to be a Discrete Normal Distribution $\mathcal{D}_{\sigma,\mathbb{Z}}$ with standard deviation $\sigma$. It is often convenient to group the $m$ LWE samples in a matrix-vector pair as follows:

$$(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m.$$

There exist several approaches to solve both SLWE and DLWE. In general, Regev gave a decision-to-search reduction for LWE, showing that SLWE and DLWE are indeed equivalent. We will discuss the approaches to solve LWE in Section 4.

### Public-key Encryption Using LWE

In this section, we report a simple public-key cryptosystem based on LWE introduced by Regev [35].

Let $\boldsymbol{s}$ chosen uniformly at random in $\mathbb{Z}_q^n$ be the private key, and let $(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ obtained from $L_{\boldsymbol{s},\chi_\sigma}$ be the public key. To encrypt one bit $x$ of the message, one does the following. Choose a random subset $S$ among the $2^m$ subsets of $[m]$, and compute the ciphertext

$$(\bar{\boldsymbol{a}}, \bar{b}) = \left( \sum_{i \in S} \boldsymbol{a}_i, x \cdot \left\lfloor \frac{q}{2} \right\rfloor + \sum_{i \in S} b_i \right).$$

In decryption, one uses the private key $\boldsymbol{s}$ and decides that $x = 0$ if $\bar{b} - \bar{\boldsymbol{a}} \cdot \boldsymbol{s}$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$, $x = 1$ otherwise.

### LWE Challenge

The *TU Darmstadt LWE Challenge* is a series of LWE instances that are left for the public to be solved [37]. These uses $c_q = 2$, $\alpha$ ranging from 0.005 to 0.07, and $n$ ranging from 40 to 120. Also, for each instance, $m = n^2$ samples are provided. The scope of this challenge is to give a tool for the cryptographers to test LWE solvers with practical instances and make comparisons among them. Among the largest instances solved so far in the challenge, there are $(n, \alpha) = (40, 0.03)$ and $(n, \alpha) = (75, 0.005)$.

### On the Secret Distribution

Several variants of LWE have been proposed differing in how the secret is sampled, i.e., according to certain non-uniform distributions. In [38], the following reduction that transforms a standard LWE instance to an LWE instance with the secret distributed according to the error distribution was introduced.

Let $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\cdot\boldsymbol{s}+\boldsymbol{e} \mod q)$ be a standard LWE instance. Compute, using Gaussian Elimination, $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{m\times n}$ such that $\bar{\boldsymbol{A}}^T$ is the systematic form of $\boldsymbol{A}^T$. Assume the first $n$ rows of $\bar{\boldsymbol{A}}$ are linearly independent and let $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times n}$ be the matrix formed by these rows. Consider the following change of variables

$$\hat{\boldsymbol{s}} = \boldsymbol{A}_0 \cdot \boldsymbol{s} - \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \qquad \hat{\boldsymbol{A}} = \boldsymbol{A} \cdot \boldsymbol{A}_0^{-1} = \begin{pmatrix} \mathbf{I}_n \\ \hat{\boldsymbol{a}}_{n+1} \\ \vdots \\ \hat{\boldsymbol{a}}_m \end{pmatrix}, \qquad \hat{\boldsymbol{b}} = \boldsymbol{b} - \hat{\boldsymbol{A}} \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \hat{b}_{n+1} \\ \vdots \\ \hat{b}_m \end{pmatrix}.$$

Then, the pair $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})$ is an LWE problem where the entries of the secret is equal to the first $n$ entries of the error vector from the original problem times $-1$. To make this transformation, one "loses" $n$ LWE samples. However, due to this reduction, this version of LWE is generally considered as hard as standard LWE.

In [39], Brakerski et al. studied another variant of LWE, generally referred to as *binary-LWE*, where the secret takes values in $\{0, 1\}$. In particular, they gave a reduction from standard LWE to binary-LWE with dimension increasing from $n$ to $n \log q$.

### Ring-LWE

The idea of defining an LWE-like problem over polynomial rings was initially studied in [40] and [41]. Nowadays, it is one of the most promising variants of LWE, thanks to the

fact that it provides smaller key-sizes and more efficient implementations than standard LWE.

Let $n$ be a power of 2. For $q$ prime, let $R$ and $R_q$ be polynomial rings defined as in Section 3.1.

**Definition 21.** (Ring-LWE) *Let $\psi_\sigma$ be a non-uniform distribution over on $R$ with standard deviation $\sigma$. Given a secret polynomial $s(x) \in R_q$ chosen uniformly at random, denote with $L^r_{s(x),\psi_\sigma}$ the probability distribution over $R_q \times R_q$ obtained by sampling $a(x) \in R_q$ uniformly at random, $e(x) \in R_q$ according to $\psi_\sigma$, and returning*

$$(a(x),\ b(x) = a(x) \star s(x) + e(x)) \in R_q \times R_q.$$

- *The Search version of the Ring-LWE problem consists of recovering the secret $s(x)$ given $m \geq 1$ samples from $L^r_{s(x),\psi_\sigma}$.*
- *The Decision version of the Ring-LWE problem consists of, given $m \geq 1$ pairs $(a(x),\ b(x)) \in R_q \times R_q$, deciding whether these are sampled from $L^r_{s(x),\psi_\sigma}$ or uniformly at random from $R_q \times R_q$.*

One can write a sample of Ring-LWE as $n$ samples of LWE where the public matrix $\boldsymbol{A}$ has a special structure and, therefore, there is a sort of correlation among its rows. Furthermore, it can be seen as the inhomogeneous version of the NTRU Key-Recovery Problem (Definition 18). In particular, it was proved to be at least as hard as NTRU [14]. Ring-LWE also benefits by a worst-case reduction from approximate SVP on ideal lattices[5] to the search version Ring-LWE [41].

## 3.3 On Using the Ring of Integers $\mathbb{Z}$

The main mathematical objects in the NTRU cryptosystem and in the cryptosystems based on Ring-LWE are polynomials in $R_q$. With these, it is often required to develop some additional ring-specific technicalities to obtain efficient implementations. Some researchers worked around this issue by introducing new cryptosystems that use, instead, integer numbers as main mathematical object resulting easy to implement. These base their security on hard problems defined on $\mathbb{Z}$, analogues to the aforementioned defined on $R_q$. In this section, we report the underlying hard problems of two cryptosystems that can be seen as the integer variants of the NTRU Key-Recovery Problem and Ring-LWE. Another important cryptosystem following this direction, not covered by this

[5] a generalization of cyclic lattices.

section, is *The Three Bears Cryptosystem* [42], a candidate to the NIST Post-Quantum Standardization at round two [12].

### The AJPS Cryptosystem

In 2018, Aggarwal et al. introduced an NTRU-like public-key cryptosystem that exploits a property of the Mersenne prime numbers (i.e., a prime of the form $2^k - 1$, for some integer $k$) and uses the elements of $\mathbb{Z}$ as main mathematical object [43]. This was accepted among the competitors for the NIST Post-Quantum Standardization at round one [12].

Let the *Hamming weight* of an integer in the range $[0, 2^k - 1]$ be the number of non-zero bits of its binary representation. The security of the AJPS cryptosystem bases its security on the assumption that the following is a hard problem.

**Definition 22.** (Mersenne Low Hamming Ratio Search Problem) *Let $q = 2^k - 1$ be a Mersenne prime number, $h < k$ an integer, $F$ and $G$ two integers chosen at random from the set of $k$-bit numbers with Hamming weight $h$. Let $H$ be an integer such that*

$$F \cdot H \equiv G \mod q. \tag{3.3}$$

*The Mersenne Low Hamming Ratio Search Problem (MLHRatioSP) is to find $(F, G)$ given $h$ and $H$.*

Beunardeau et al. showed how to perform a lattice-based attack on this problem [44]. Such attack was later analyzed in [45]. This requires more technicalities compared to the attack in standard NTRU.

### Integer Ring-LWE

A variant of Ring-LWE on the ring of integers $\mathbb{Z}$, called *Integer Ring-LWE* was introduced by Gu [46].

Let $q, n$ be two positive integers such that $q$ is prime and $q > n^3$, and let $p = q^n + 1$. Given an error distriubtion $\psi_\sigma$ over $\mathbb{Z}$, let $\psi^n_{\sigma,q}$ be the probability distribution that samples $u_0, ...., u_{n-1}$ at random from $\psi_\sigma$ and returns $u = \sum_{i=0}^{n-1} u_i q^i \in \mathbb{Z}_p$.

**Definition 23.** (Integer Ring-LWE) *Given a secret positive integer $s \in \mathbb{Z}_p$ sampled at random from $\psi^n_{\sigma,q}$, denote with $L^i_{s,\psi_\sigma}$ the probability distribution over $\mathbb{Z}_p \times \mathbb{Z}_p$ obtained*

*by sampling $a \in \mathbb{Z}_p$ uniformly at random, $e \in \mathbb{Z}_p$ from $\psi^n_{\sigma,q}$, and returning*

$$(a, b = a \cdot s + e \bmod p) \in \mathbb{Z}_p \times \mathbb{Z}_p.$$

- *The Search version of the Integer Ring-LWE problem consists of, given $m \geq 1$ samples from $L^i_{s,\psi_\sigma}$, recovering the secret $s$.*
- *The Decision version of the Integer Ring-LWE problem consists of, given $m \geq 1$ pairs $(a,\ b) \in \mathbb{Z}_p \times \mathbb{Z}_p$, deciding whether these are sampled from $L^i_{s,\psi_\sigma}$ or uniformly at random from $\mathbb{Z}_p \times \mathbb{Z}_p$.*

We refer the reader to [46] and [47] (in which the problem is called *Integer Polynomial Learning With Errors*) for security reductions regarding Integer Ring-LWE.

# 4 Attacks to LWE

Due to its relevancy in Post-Quantum Cryptography, the Learning With Errors problem has been widely studied and cryptanalysed in recent years. In this section, we survey the three main approaches discovered so far to solve LWE. For a detailed analysis of the asymptotic complexity of solving LWE, we refer the reader to [48].

## 4.1 Algebraic Attacks

Arora and Ge proposed an approach based on algebraic methods to solve LWE instances [49]. The main idea consists of writing a system of equations such that its resolution implies solving an LWE problem instance.

Let $(\boldsymbol{a}, b = \boldsymbol{a} \cdot \boldsymbol{s} + e)$ be one LWE sample, and let $D \subset \mathbb{Z}_q$ be a set such that $e \in D$ with high probability. For example, in case of a Discrete Normal Distribution, one can take $D = \{-U\sigma, ..., +U\sigma\}$, for a small positive integer $U$. In case of LWE with *binary error*, one takes $D = \{0, 1\}$. Consider the following polynomial of $\mathbb{Z}_q[\boldsymbol{x}] = \mathbb{Z}_q[x_1, ..., x_n]$

$$p(\boldsymbol{x}) = \prod_{i \in D} (b - \boldsymbol{a} \cdot \boldsymbol{x} - i).$$

Then, the secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ is a *zero* of $p(\boldsymbol{x})$, i.e., $p(\boldsymbol{s}) = 0$. Also, most likely, one has that $p(\boldsymbol{z}) \neq 0$, for $\boldsymbol{z} \neq \boldsymbol{s}$. Given $m$ LWE samples, the system of polynomial equations

$$\{p_i(\boldsymbol{x}) = 0\}_{i=0}^{m} \tag{4.1}$$

has $\boldsymbol{s}$ as a unique solution with high probability. Therefore, solving (4.1) corresponds to a secret-recovery of the LWE problem instance. To do so, Arora and Ge proposed to *linearize* the system by replacing each monomial with a new variable and then use *Gaussian elimination* to solve it. Since the number of monomials is

$$\begin{pmatrix} n+ \mid D \mid \\ \mid D \mid \end{pmatrix},$$

the complexity of this attack has a strict dependency on the size of $D$. Albrecht et al. improved the approach by employing Gröbner bases to solve (4.1) instead of linearization [50]. In general, this algorithm is subexponential for very small and bounded error. In particular, given $m \geq 6.6n$ samples of an LWE instance with binary error in dimension $n$, one can solve it in time $n^2 2^{0.334n}$ and memory $n^2 2^{0.289n}$ [50]. However, for larger noise, this method becomes inefficient.

## 4.2 Combinatorial Attacks

Informally, the so-called *Learning Parity with Noise* (LPN) problem can be seen as a particular case of LWE for $q = 2$. Blum, Kalai, and Wasserman introduced the so-called BKW algorithm as a sub-exponential algorithm to solve LPN [51]. Later, Albrecht et al. presented the first adaptation of the BKW algorithm to LWE [52].

The main idea of the BKW algorithm is to combine the samples with sums and subtractions to obtain a new, easier-to-solve LWE problem having the same secret. Then, one applies an efficient guessing procedure for solving it.

Consider an LWE problem instance in dimension $n$, with secret and error sampled from a Discrete Normal Distribution $\mathcal{D}_{\sigma,\mathbb{Z}}$ in matrix form

$$(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m.$$

Let $\boldsymbol{a}_{i_1}$ and $\boldsymbol{a}_{i_2}$ be two rows of $\boldsymbol{A}$ that have in common the first $t < n$ entries[6]. Then

$$\boldsymbol{a}' = \boldsymbol{a}_{i_1} - \boldsymbol{a}_{i_2} = (\underbrace{0, ..., 0}_{t \text{ entries}}, \tilde{\boldsymbol{a}}) \mod q, \qquad \tilde{\boldsymbol{a}} \in \mathbb{Z}_q^{n-t}.$$

Let $b' = b_{i_1} - b_{i_2} \mod q$ and $e' = e_{i_1} - e_{i_2} \mod q$. Then, the pair

$$(\boldsymbol{a}',\ b' = \boldsymbol{a}' \cdot \boldsymbol{s} + e' \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

is a sample of an LWE problem with error distribution approximately $\mathcal{D}_{\sqrt{2}\sigma,\mathbb{Z}}$. By performing this kind of operation, generally referred to as a *BKW step*, for all rows of $\boldsymbol{A}$ and the corresponding entries of the vector $\boldsymbol{b}$, one obtains a new LWE instance

$$(\boldsymbol{A}',\ \boldsymbol{b}' = \boldsymbol{A}' \cdot \boldsymbol{s} + \boldsymbol{e}' \mod q) \in \mathbb{Z}_q^{m' \times n} \times \mathbb{Z}_q^{m'}, \tag{4.2}$$

where the first $t$ columns of $\boldsymbol{A}'$ are set to zero. Because of this, the corresponding first $t$ entries of $\boldsymbol{s}$ play no role in (4.2). Therefore, (4.2) can be seen as an instance of an LWE problem in dimension $n - t$.

By sequentially performing several BKW steps, one can reduce the dimension of the problem to be small enough to allow an efficient error-recovery using techniques based, for example, on the Fast Fourier Transform (FFT). Since at every step the variance of the error distribution doubles, it is crucial that the final noise is still small enough to allow the recovery of the secret.

[6]Equivalently, if they have $t < n$ entries that are opposite to each other modulo $q$, one computes $\boldsymbol{a}_{i_1} + \boldsymbol{a}_{i_2} \mod q$

A general drawback of the BKW algorithm is that, as one would expect, it requires a large number of samples ($m \gg n$). A work around, known as *sample amplification*, was introduced in [53] for the case of LPN, and consists in combining the samples to obtain more samples.

Later variants of the algorithm exploited the idea of relaxing the reduction to get a better time and memory complexity [54, 55, 56, 57]. In these cases, is not required anymore the entries of the matrix $\boldsymbol{A}$ to be reduced to zero, but instead, they can be reduced to be "small".

The asymptotic complexity of the basic version of BKW for LWE was analysed in [52] and [56]. Let $\sigma = \mathcal{O}(n^{c_s})$ and $q = \mathcal{O}(n^{c_q})$. Then, LWE can be solved via BKW in time $2^{\gamma_1(n)+o(n)}$ if $m = 2^{\Theta(n)}$, and in time $2^{\gamma_2(n)+o(n)}$ if $m = \Theta(n \log n)$, where

$$\gamma_1(n) = \frac{1}{2}\frac{c_q}{c_q - c_s + 1/2}n \quad \text{and} \quad \gamma_2(n) = \frac{1}{2}\frac{c_q}{c_q - c_s}n,$$

and in memory $2^{\Theta(n)}$. For the asymptotic complexity of more developed versions of BKW, we refer the reader to [48] and [58].

## 4.3 Lattice-based Attacks

Lattice reduction is considered the best approach to solve LWE. One clear advantage compared to, for example, the BKW algorithm is that, generally, it does not require many samples to be executed. Thanks to the asymptotic analysis on the BKZ algorithm, it is possible to get security estimates on LWE parameters' choices based on these kinds of attacks. Furthermore, implementations of BKZ with enumeration or sieving [59, 30] solved the largest actual LWE instances of the TU Darmstadt LWE Challenge [37].

Two main approaches use lattice reduction to solve LWE: the *primal* attack, which addresses the Search version of LWE, and the *dual* attack, which addresses the Decision version.

Let $(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n$ be an LWE instance where the entries of $\boldsymbol{s}$ and $\boldsymbol{e}$ are sampled independently from $\mathcal{D}_{\sigma,\mathbb{Z}}$.

### Primal Attack

The strategy is to convert the LWE instance into a unique-SVP instance and then solve it using a lattice reduction algorithm like BKZ. This approach was studied in [60].

Consider the lattice

$$\Lambda = \left\{ \boldsymbol{x} \in \mathbb{Z}^{m+n+1} : (\boldsymbol{A}|I_m| - \boldsymbol{b})\boldsymbol{x} = 0 \mod q \right\},$$

where $(\boldsymbol{A}|I_m| - \boldsymbol{b})$ denotes the horizontal concatenation matrix of $\boldsymbol{A}$, $I_m$, and $-\boldsymbol{b}$. The dimension of $\Lambda$ is $d = m + n + 1$, and the volume is $q^m$. For small enough $\sigma$, the vector $\boldsymbol{v} = (\boldsymbol{s}, \boldsymbol{e}, 1)$ of norm $\lambda \approx \sigma\sqrt{n+m}$ is a unique shortest vector of $\Lambda$ and therefore a unique-SVP solution for $\Lambda$.

We refer the reader to [61] for a recent analysis of the success probability.

### Dual Attack

The idea of this attack is to use lattice reduction to find several short vectors in a lattice, and use them to distinguish an LWE pair from a random one [62, 63].

Define the lattice

$$\Lambda' = \{(\boldsymbol{c}_1, \boldsymbol{c}_2) \in \mathbb{Z}^m \times \mathbb{Z}^n : \boldsymbol{c}_1 \cdot \boldsymbol{A} = \boldsymbol{c}_2 \mod q\}.$$

In particular, the dimension and the volume of $\Lambda$ are $m+n$ and $q^n$ respectively. One uses BKZ to reduce a basis of $\Lambda'$ and find a short vector $(\boldsymbol{v}, \boldsymbol{w})$ to be used as a distinguisher for LWE. Let $\ell = \|(\boldsymbol{v}, \boldsymbol{w})\|$, then

$$\boldsymbol{y} = \boldsymbol{v} \cdot \boldsymbol{b} = \boldsymbol{v} \cdot \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{v} \cdot \boldsymbol{e} = \boldsymbol{w} \cdot \boldsymbol{s} + \boldsymbol{v} \cdot \boldsymbol{e} \mod q$$

is distributed according to $D_{\ell\sigma,\mathbb{Z}}$ if $(\boldsymbol{A}, \boldsymbol{b})$ is an LWE instance, uniform in $\mathbb{Z}_q$ otherwise. These two distributions have statistical distance upper bounded by $\varepsilon = 2\exp(-2\pi^2\tau^2)$ [64], where $\tau = \ell\sigma/q$. In order to have a reasonable success probability, one must find $1/\varepsilon^2$ vectors of norm $< \ell$. This can be done by running the BKZ algorithm the necessary number of times on different randomized bases of the lattice.

Hybrid versions of this attack combining the above with a bruteforce search, were proposed, for example, in [65, 66].

# 5 Pairing-based Cryptography

In this section, we give a brief introduction to the field of Pairing-based Cryptography, with the scope of explaining the concept of hash maps to the pairing's domain groups. We refer the reader to [67] and [68] for complete and detailed surveys on this field.

## 5.1 The Group of Points of an Elliptic Curve

**Definition 24.** (Elliptic Curve) *Let $\mathbb{F}_q$ be a finite field of characteristic $p \neq 2, 3$ and order $q = p^m$, for some $m > 0$. An elliptic curve $E$ defined over $\mathbb{F}_q$ is the graph of an equation of the form*

$$y^2 = x^3 + Ax + B,$$

*for $A, B \in \mathbb{F}_q$ such that $4A^3 + 27B^2 \neq 0$.*

**Definition 25.** (Supersingular Elliptic Curve) *Let $\mathbb{F}_q$ be a prime field. An elliptic curve $E$ defined over $\mathbb{F}_q$ is said to be supersingular if $\#E(\mathbb{F}_q) = q + 1$. Otherwise, it is said to be ordinary.*

Let $\bar{\mathbb{F}}_q \supseteq \mathbb{F}_q$ be the algebraic closure of $\mathbb{F}_q$. The set $E(\bar{\mathbb{F}}_q)$ of all points of $E$ taking values in $\bar{\mathbb{F}}_q$, together with a so-called *point at infinity* $\mathcal{O}$, is an abelian group with respect to a specific sum operation denoted with $\oplus$ (see [67, Sec. 2.2]). In particular, $\mathcal{O}$ is the identity element of the group, that is $P \oplus \mathcal{O} = \mathcal{O} \oplus P = P$. We denote with $-P$ the point of the elliptic curve symmetrical to $P$ respect to the $x-$axis. Furthermore, given a positive integer $a$, we define the multiplication of a point $P$ by a scalar $c$ as

$$cP := \underbrace{P \oplus P \oplus ... \oplus P}_{c \text{ times}}.$$

**Definition 26.** (Embedding Degree) *Let $E$ be an Elliptic curve defined over $\mathbb{F}_q$ and $\mathbb{G}$ be a subgroup of $E(\bar{F}_q)$ of order $r$. Let $k$ be the smallest integer such that $r$ divides $q^k - 1$, then $\mathbb{G}$ is said to have embedding degree $k$.*

**Definition 27.** ($r$-torsion Group) *Let $E$ be an elliptic curve defined over a finite field $\mathbb{F}_q$ and let $r$ be a positive integer. The $r$-torsion group of $E$, denoted by $E[r]$, is defined as the set of all points of $E$ of order $r$, i.e.*

$$E[r] = \{P \in E(\bar{\mathbb{F}}_q) \mid rP = \mathcal{O}\}$$

The idea of using elliptic curves in Cryptography was introduced independently by

Miller in 1986 and Koblitz in 1987 [69, 70]. The following is the main hard computational problem on which Elliptic Curve Cryptography is based on.

**Definition 28.** (Elliptic Curve Discrete Logarithm Problem (ECDLP)) *Let $E$ be an elliptic curve defined over a finite field $\mathbb{F}_q$. Let $\mathbb{G} \subseteq E(\bar{\mathbb{F}}_q)$ be a group of prime order $r$, and $P$ be a generator for $\mathbb{G}$. The Elliptic Curve Discrete Logarithm Problem consists of, given a point $Q \in \mathbb{G}$, finding $n \in \mathbb{Z}_r$ such that $Q = nP$.*

## 5.2 Pairings in Cryptography

We now introduce the definition of the main mathematical object which characterizes Pairing-based Cryptography. Pairings were initially employed in Cryptography as a tool to build attacks to ECDLP. More specifically, they were used to reduce ECDLP to a classical Discrete Logarithm Problem (DLP) over finite fields [71, 72]. Later on, researchers found out how to use them to build cryptographic protocols covering a wide variety of features [73, 74, 75].

**Definition 29.** (Pairing) *A pairing is a map $e$ defined on the direct product of two abelian, finite and additive cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to some other abelian, finite and multiplicative cyclic group $\mathbb{G}_T$*

$$e(\cdot, \cdot) : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

*which has the following properties:*

1. *Bilinear: for any pair $(P_1, P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ and for any $(a, b) \in \mathbb{Z} \times \mathbb{Z}$, we have $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$.*
2. *Non-degenerate: if $e(P, S) = 1$ for any $S \in \mathbb{G}_2$, then $P = \mathcal{O}$, and if $e(P, S) = 1$ for any $P \in \mathbb{G}_1$, then $S = \mathcal{O}$.*
3. *Computable: there is a polynomial-time algorithm to compute $e(P_1, P_2)$, for any $(P_1, P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$.*

The Weil pairing and the Tate pairing are examples of pairings where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of points of an elliptic curve. More in general, pairings used in Cryptography are classified into four types according to the choice of $\mathbb{G}_1$ and $\mathbb{G}_2$, usually taken as subgroups of $E[r]$.

- **Type 1**. The elliptic curve is supersingular and $\mathbb{G}_1 = \mathbb{G}_2$.

- **Type 2**. The elliptic curve is ordinary, and $\mathbb{G}_1 \neq \mathbb{G}_2$. There exists an efficiently computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.
- **Type 3**. The elliptic curve is ordinary, and $\mathbb{G}_1 \neq \mathbb{G}_2$. There exists an homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, but there is no known efficient way to compute it.
- **Type 4**. The elliptic curve is ordinary, and $\mathbb{G}_1 \neq \mathbb{G}_2$. In this case $\mathbb{G}_2 = E[r]$.

For efficiency and security reasons, type 3 pairings are the ones used in practice. In this case, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are chosen to be

$$\mathbb{G}_1 = E(\mathbb{F}_q) \cap E[r], \quad \mathbb{G}_2 = E[r] \cap \ker(\pi - [q]), \quad \mathbb{G}_T = (\mathbb{F}_{q^k})^*,$$

where $\pi$ is the *Frobenius endomorphism*, $[q] : E(\mathbb{F}_q) \to E(\mathbb{F}_q)$ maps $P \mapsto qP$, $r$ is a prime divisor of $\#E(\mathbb{F}_q)$ and $k$ is the corresponding embedding degree.

For security reasons, one wants that both ECDLP and DLP are hard respectively in $E(\mathbb{F}_q)$ and $(\mathbb{F}_{q^k})^*$. The elliptic curves that satisfy such conditions, for a relatively small $k$, and that are suitable for use in combination with pairings are called *pairing-friendly*. It has been shown that these kinds of curves are rare, and they require dedicated constructions [76, 77]. The MNT curves were the first family of pairing-friendly elliptic curves to be discovered [78]. Then, several other families have been proposed [79, 80, 81, 82, 83].

### Joux's Three Party Key Agreement

To give an example of use of pairings in Cryptography, we report the key agreement protocol introduced by Joux [73]. For simplicity, we assume to use a type-1 pairing that is also symmetric. However, it is easy to generalize it for type-3 pairings.

Let Alice, Bob, and Chris be three participants that need to agree on a secret (key). Let $P \in \mathbb{G}_1$ be a point publicly known and let $a, b$, and $c$ be three secret scalars chosen at random by the three participant respectively. Alice computes $aP$ and sends it over the public channel. She receives $bP$ and $cP$ from Bob and Chris (see Figure 5). Alice can compute a shared key as

$$k = e(bP, cP)^a = e(P, P)^{abc}.$$

Similarly, Bob computes $k = e(aP, cP)^b$ and Chris $k = e(aP, bP)^c$. All the three participants agreed on a common secret $k$.

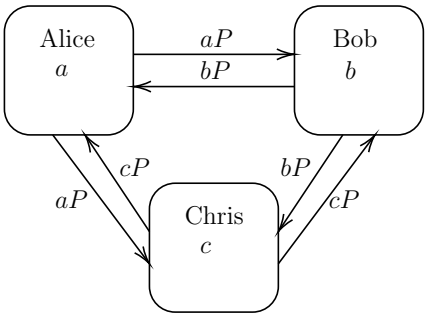


Figure 5: Joux's three-party key agreement

## 5.3 Hash-Maps to $\mathbb{G}_1$ and $\mathbb{G}_2$

Several protocols supporting *Identity-based Encryption* (IBE) have been constructed using pairings. This type of public-key encryption, conceived for the first time by Shamir in 1985 [84], uses some unique information directly linked with the identity of a user (e.g., name, phone number, email address, IP address, etc.) as a public key.

Typically, given a binary string $ID$ representative of a user, one wants to map (hash) such a string into a mathematical object to be used within a protocol, namely a point of $\mathbb{G}_1$ or $\mathbb{G}_2$. Let $\{0,1\}^*$ be the set of all binary strings of any fixed length. With $H_1$ and $H_2$, we denote two *hash-maps* that map an alphanumeric string (e.g., a phone number), written in its binary notation as an element $ID \in \{0,1\}^*$, respectively to $\mathbb{G}_1$ and $\mathbb{G}_2$:

$$\begin{array}{cc} H_1 : \{0,1\}^* \to \mathbb{G}_1, & H_2 : \{0,1\}^* \to \mathbb{G}_2 \\ ID \mapsto H_1(ID), & ID \mapsto H_2(ID). \end{array}$$

For the case of type 3 pairings, computing $H_1$ can be done efficiently using the so-called *cofactor multiplication method*. The group $\mathbb{G}_1$ is the unique subgroup of order $r$ of $E(\mathbb{F}_q)$. One first hashes $ID$ to a random point $P \in E(\mathbb{F}_q)$, then multiplies it by the cofactor $c = \#E(\mathbb{F}_q)/r$. In this way, we are sure that $cP \in \mathbb{G}_1$. Theoretically, this same approach can be applied for hashing to $\mathbb{G}_2$ too. However, the cofactor of $\mathbb{G}_2$ is generally a very large integer which makes this multiplication expensive. Scott *et al.* [85] proposed a method to speed up such multiplication. A further improvement of this method was later obtained by Fuentes *et al.* [86].

# 6 Contributions

In this section, we list the contribution to the field for each paper included in this thesis.

In [87], we construct a non-trivial reduction of the Mersenne Low Hamming Combination Search Problem (MLHCombSP)[7] to Integer Linear Programming (ILP). This reduction allows to build a new framework to solve MLHCombSP. Furthermore, we discovered a new family of weak keys for which a key-recovery attack is possible and runs in polynomial time.

In [88], we make a concrete analysis of the complexity of Integer-RLWE. In particular, we provide the details to perform a the meet-in-the-middle attack and a lattice-based attack for the case of Integer-RLWE. Then, we exploit a weakness on the parameters given by Gu [46] to build a more efficient lattice-based attack. This drastically reduces the security of Integer-RLWE for weak choices of $n$.

In [89], we present two main improvements for the BKW algorithms: a new reduction step that significantly improves the so-called Lazy Modulus Switching technique introduced in [54], and a new guessing method based on the Fast Walsh Hadamard Transform (FWHT). We present two implementations of BKW for LWE that can solve relatively large instances of the problem. The first one, called RBBL (RAM-Based BKW for LWE), is faster and uses only RAM memory. The second one, called FBBL (File-Based BKW for LWE), uses a combination of RAM and disk memory to overcome the limitations of RBBL and solve larger instances. With these implementations, we solve some LWE problems with a large number of samples and with parameters as in the TU Darmstadt LWE Challenge [37]. Furthermore, we use sample amplification to solve the original TU Darmstadt LWE Challenge with parameters $(n, \alpha) = (40, 0.005)$ for which only $n^2$ samples are provided.

Since the publication of Kim and Barbulescu's attack on the Discrete Logarithm Problem in finite field extensions [90], the BLS family of pairing-friendly elliptic curves [79] gained interest in the community and found widespread adoption in applications. In [91], we fill a gap in the field and compute explicit formulas to hash to $\mathbb{G}_2$ for the BLS family of pairing-friendly elliptic curves. In particular, we apply both Scott et al. [85] and Fuentes et al. [86] methods and compare the maps obtained with the two for embedding degree $k \in \{12, 24, 30\}$. For the cases of $k = 42$ and $k = 48$, we could not apply Fuentes et al. method directly, but we use theoretical means to obtain hash maps that are more efficient than the one obtained with the Scott et al. method.

[7] the inhomogeneous version of MLHRatioSP (see Definition 22).

# Bibliography

[1] A. Kerckhoffs, "La Cryptographie Militaire," *Journal des Sciences Militaires*, pp. 161–191, 1883.

[2] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[3] R. F. Churchhouse, *Codes and Ciphers: Julius Caesar, the Enigma, and the Internet*. Cambridge University Pres, 2001.

[4] D. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996.

[5] J. H. Ellis, "The Possibility of Secure Non-Secret Digital Encryption." https://cryptocellar.org/cesg/possnse.pdf, 1970.

[6] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transaction on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A Method For Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[8] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[9] P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[10] F. Arute et al., "Quantum Supremacy using a Programmable Superconducting Processor," *Nature*, vol. 574, pp. 505–510, 2019.

[11] H. Zhong et al., "Quantum Computational Advantage using Photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020.

[12] NIST, "Post-quantum cryptography standardization." https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[13] S. Lang, *Algebra*. Springer, 2002.

[14] *A Decade of Lattice Cryptography*, vol. 10, Now Publishers, 2016.

[15] S. D. Galbraith, *Mathematics of Public Key Cryptography*. https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html, 2nd ed., 2018.

[16] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*. Springer, 2nd ed., 2014.

[17] P. Q. Nguyen and B. Valle, *The LLL Algorithm: Survey and Applications*. Springer, 1st ed., 2009.

[18] C. Hermite, "Extraits de Lettres de M. Ch. Hermite à M. Jacobi sur Différents Oobjects de la Théorie des Nombres. (Continuation).," *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1850, no. 40, pp. 279–315, 1850.

[19] J. H. Conway, N. J. A. Sloane, and E. Bannai, *Sphere-Packings, Lattices, and Groups*. Springer, 1987.

[20] J. Milnor and D. Husemoller, *Symmetric Bilinear Forms*. Springer, 1973.

[21] A. K. Lenstra, H. W. Lenstra, and L. Lovasz, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[22] C. Schnorr, "A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms," vol. 53, pp. 201–224, Elsevier, 1987.

[23] C. Schnorr and M. Euchner, "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems," *Mathematical Programming*, vol. 66, pp. 181–199, 08 1994.

[24] G. Hanrot, X. Pujol, and D. Stehlé, "Terminating BKZ." Cryptology ePrint Archive, Report 2011/198, 2011. https://eprint.iacr.org/2011/198.

[25] N. Gama, P. Q. Nguyen, and O. Regev, "Lattice Enumeration Using Extreme Pruning," in *Advances in Cryptology – EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 257–278, Springer, 2010.

[26] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better Lattice Security Estimates," in *Advances in Cryptology – ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 1–20, Springer, 2011.

[27] T. Laarhoven, "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing," in *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *LNCS*, pp. 3–22, Springer, 2015.

[28] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 10–24, SIAM, 2016.

[29] T. Laarhoven, "Search Problems in Cryptography." PhD thesis, Eindhoven University of Technology, 2015. https://thijs.com/docs/phd-final.pdf.

[30] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in *Advances in Cryptology – EUROCRYPT 2019*, vol. 11477 of *LNCS*, pp. 717–746, Springer, 2019.

[31] J. Neyman and E. S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, pp. 289–337, 1933.

[32] G. Casella and R. Berger, *Statistical Inference*. Duxbury Advanced Series in Statistics and Decision Sciences, Thomson Learning, 2002.

[33] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem," in *Algorithmic Number Theory, LNCS*, vol. 1423, pp. 267–288, Springer, 1998.

[34] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, P. Schwabe, W. Whyte, and Z. Zhang, "NTRU: Algorithm Specifications And Supporting Documentation." https://ntru.org/f/ntru-20190330.pdf, 2019.

[35] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pp. 84–93, ACM, 2005.

[36] C. Peikert, "Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: Extended Abstract," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 333–342, ACM, 2009.

[37] "TU Darmstadt Learning with Errors Challenge." https://www.latticechallenge.org/lwe_challenge/challenge.php. Accessed: 2021-10-02.

[38] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems," in *Advances in Cryptology - CRYPTO 2009*, vol. 5677 of *LNCS*, pp. 595–618, Springer, 2009.

[39] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical Hardness of Learning with Errors," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 575–584, ACM, 2013.

[40] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient Public-Key Encryption Based on Ideal Lattices," in *Advances in Cryptology – ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 617–635, Springer, 2009.

[41] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Advances in Cryptology – EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 1–23, Springer, 2010.

[42] M. Hamburg, "Module-LWE Key Exchange and Encryption: The Three Bears." https://www.shiftleft.org/papers/threebears/threebears-draft1.pdf, 2017.

[43] D. Aggarwal, A. Joux, A. Prakash, and M. Santha, "A New Public-Key Cryptosystem via Mersenne Numbers," in *Advances in Cryptology – CRYPTO 2018*, vol. 10993 of *LNCS*, pp. 459–482, Springer, 2018.

[44] M. Beunardeau, A. Connolly, R. Géraud, and D. Naccache, "On the Hardness of the Mersenne Low Hamming Ratio Assumption," in *Progress in Cryptology – LATINCRYPT 2017*, vol. 11368 of *LNCS*, pp. 166–174, Springer, 2019.

[45] K. de Boer, L. Ducas, S. Jeffery, and R. de Wolf, "Attacks on the AJPS Mersenne-Based Cryptosystem," in *Post-Quantum Cryptography*, vol. 10786 of *LNCS*, pp. 101–120, Springer, 2018.

[46] C. Gu, "Integer Version of Ring-LWE and Its Applications," in *Security and Privacy in Social Networks and Big Data - 5th International Symposium, SocialSec 2019, Copenhagen, Denmark*, vol. 1095 of *Communications in Computer and Information Science*, pp. 110–122, Springer, 2019.

[47] J. Devevey, A. Sakzad, D. Stehlé, and R. Steinfeld, "On the Integer Polynomial Learning with Errors Problem," in *Public-Key Cryptography – PKC 2021*, vol. 12710 of *LNCS*, pp. 184–214, Springer, 2021.

[48] G. Herold, E. Kirshanova, and A. May, "On the Asymptotic Complexity of Solving LWE," *Designs, Codes and Cryptography*, vol. 86, pp. 55–83, 2018.

[49] S. Arora and R. Ge, "New Algorithms for Learning in Presence of Errors," in *Automata, Languages and Programming*, vol. 6755 of *LNCS*, pp. 403–415, Springer, 2011.

[50] M. R. Albrecht, C. Cid, J. C. Faugère, R. Fitzpatrick, and L. Perret, "Algebraic Algorithms for LWE Problems," in *ACM Communications in Computer Algebra*, vol. 49, pp. 62–91, ACM, 2015.

[51] A. Blum, A. Kalai, and H. Wasserman, "Noise-Tolerant Learning, the Parity Problem, and the Statistical Query model," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pp. 435–440, ACM, 2000.

[52] M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, and L. Perret, "On the Complexity of the BKW Algorithm on LWE," *Design, Codes and Cryptography*, vol. 74, no. 2, pp. 325–354, 2015.

[53] V. Lyubashevsky, "The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem," in *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, vol. 2129 of *LNCS*, pp. 378–389, Springer, 2005.

[54] M. R. Albrecht, J.-C. Faugère, R. Fitzpatrick, and L. Perret, "Lazy Modulus Switching for the BKW Algorithm on LWE," in *Proceedings of the 17th International Conference on Public-Key Cryptography – PKC 2014*, vol. 8383 of *LNCS*, pp. 429–445, Springer, 2014.

[55] Q. Guo, T. Johansson, and P. Stankovski, "Coded-BKW: Solving LWE Using Lattice Codes," in *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *LNCS*, pp. 23–42, Springer, 2015.

[56] P. Kirchner and P. Fouque, "An Improved BKW Algorithm for LWE with Applications to Cryptography and lattices," in *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *LNCS*, pp. 43–62, Springer, 2015.

[57] Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski, "Coded-BKW with Sieving," in *Advances in Cryptology – ASIACRYPT 2017*, vol. 10624 of *LNCS*, pp. 323–346, Springer, 2017.

[58] Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski Wagner, "On the Asymptotics of Solving the LWE Problem Using Coded-BKW With Sieving," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 5243–5259, 2019.

[59] R. Xu, S. L. Yeo, K. Fukushima, T. Takagi, H. Seo, S. Kiyomoto, and M. Henricksen, "An Experimental Study of the BDD Approach for the Search LWE Problem," in *Applied Cryptography and Network Security - 15th International Conference*, vol. 10355 of *LNCS*, pp. 253–272, Springer, 2017.

[60] M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, "On the Efficacy of Solving LWE by Reduction to Unique-SVP," in *Information Security and Cryptology – ICISC 2013*, vol. 8565 of *LNCS*, pp. 293–310, Springer, 2014.

[61] E. W. Postlethwaite and F. Virdia, "On the Success Probability of Solving Unique SVP via BKZ," in *Public-Key Cryptography – PKC 2021*, vol. 12710 of *LNCS*, pp. 68–98, Springer, 2021.

[62] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*, pp. 147–191, Springer, 2009.

[63] M. R. Albrecht, "On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL," in *Advances in Cryptology – EUROCRYPT 2017*, vol. 10212 of *LNCS*, pp. 103–129, Springer, 2017.

[64] D. Micciancio and O. Regev, "Worst-Case to Average-case Reductions Based on Gaussian Measures," in *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 372–381, IEEE, 2004.

[65] T. Espitau, A. Joux, and N. Kharchenko, "On a Dual/Hybrid Approach to Small Secret LWE," in *Progress in Cryptology – INDOCRYPT 2020*, vol. 12578 of *LNCS*, pp. 440–462, Springer, 2020.

[66] L. Bi, X. Lu, J. Luo, K. Wang, and Z. Zhang, "Hybrid Dual Attack on LWE with Arbitrary Secrets." Cryptology ePrint Archive, Report 2021/152, 2021. https://ia.cr/2021/152.

[67] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography, Second Edition*. Chapman & Hall/CRC, 2008.

[68] N. E. Mrabet and M. Joye, *Guide to Pairing-Based Cryptography*. Chapman & Hall/CRC, 2016.

[69] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Advances in Cryptology — CRYPTO 1985*, vol. 218 of *LNCS*, pp. 417–426, Springer, 1986.

[70] N. Koblitz, "Elliptic Curve Cryptosystems," in *Mathematics of Computation*, vol. 48, pp. 203–209, AMS, 1987.

[71] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field," in *IEEE Transactions on Information Theory*, vol. 39, pp. 1639–1646, IEEE, 1993.

[72] G. Frey and H.-G. Rück, "A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves," in *Mathematics of Computation*, vol. 62, pp. 865–874, American Mathematical Society, 1994.

[73] A. Joux, "A One Round Protocol for Tripartite Diffie–Hellman," in *Algorithmic Number Theory*, vol. 1838 of *LNCS*, pp. 385–393, Springer, 2000.

[74] E. Okamoto and T. Okamoto, "Cryptosystems Based on Elliptic Curve Pairing," in *Modeling Decisions for Artificial Intelligence*, vol. 3558 of *LNCS*, pp. 13–23, Springer, 2005.

[75] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," in *Advances in Cryptology — CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 213–229, Springer, 2001.

[76] R. Balasubramanian and N. Koblitz, "The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes–Okamoto–Vanstone Algorithm," in *Journal of Cryptology*, vol. 11, pp. 141–145, Springer, 1998.

[77] F. Luca, D. Mireles, and I. E. Shparlinski, "MOV Attack in Various Subgroups on Elliptic Curves," in *Illinois Journal of Mathematics*, vol. 48, pp. 1041–1052, Duke University Press, 2004.

[78] A. Miyaji, M. Nakabayashi, and S. Nonmembers, "New Explicit Conditions of Elliptic Curve Traces for FR-Reduction," in *Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 84, IEICE, 2001.

[79] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing Elliptic Curves with Prescribed Embedding Degrees," in *Security in Communication Networks*, vol. 2576 of *LNCS*, pp. 257–267, Springer, 2003.

[80] P. S. L. M. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order," in *Selected Areas in Cryptography*, vol. 3897 of *LNCS*, pp. 319–331, Springer, 2006.

[81] D. Freeman, "Constructing Pairing-Friendly Elliptic Curves with Embedding Degree 10," in *Algorithmic Number Theory*, vol. 4076 of *LNCS*, pp. 452–465, Springer, 2006.

[82] E. J. Kachisa, E. F. Schaefer, and M. Scott, "Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field," in *Pairing-Based Cryptography – Pairing 2008*, vol. 5209 of *LNCS*, pp. 126–135, Springer, 2008.

[83] M. Scott and A. Guillevic, "A New Family of Pairing-Friendly Elliptic Curves," in *Arithmetic of Finite Fields*, vol. 11321 of *LNCS*, pp. 43–57, Springer, 2018.

[84] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in *Advances in Cryptology*, vol. 196 of *LNCS*, pp. 47–53, Springer, 1985.

[85] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and E. J. Kachisa, "Fast Hashing to G2 on Pairing-Friendly Curves," in *Pairing-Based Cryptography – Pairing 2009*, vol. 5671 of *LNCS*, pp. 102–113, Springer, 2009.

[86] L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez, "Faster Hashing to G2," in *Selected Areas in Cryptography*, vol. 7707 of *LNCS*, pp. 412–430, Springer, 2012.

[87] A. Budroni and A. Tenti, "The Mersenne Low Hamming Combination Search Problem Can Be Reduced to an ILP Problem," in *Progress in Cryptology – AFRICACRYPT 2019*, vol. 11627 of *LNCS*, pp. 41–55, Springer, 2019.

[88] A. Budroni, B. Chetioui, and E. Franch, "Attacks on Integer-RLWE," in *Proceeding of the 22nd International Conference on Information and Communications Security, 2020*, vol. 12282 of *LNCS*, pp. 528–542, Springer, 2020.

[89] A. Budroni, Q. Guo, T. Johansson, E. Mårtensson, and P. S. Wagner, "Improvements on Making BKW Practical for Solving LWE," in *Special Issue "Public-Key Cryptography in the Post-quantum Era" of Cryptography*, vol. 5, MDPI, 2021.

[90] T. Kim and R. Barbulescu, "Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case," in *Advances in Cryptology – CRYPTO 2016*, vol. 9814 of *LNCS*, pp. 543–571, Springer, 2016.

[91] A. Budroni and F. Pintore, "Efficient Hash Maps to G2 on BLS Curves," *Applicable Algebra in Engineering, Communication and Computing*, 2020.

# Part II

# Included Papers

# Paper 1

# The Mersenne Low Hamming Combination Search Problem can be reduced to an ILP Problem

*Alessandro Budroni and Andrea Tenti*

**Abstract:** In 2017, Aggarwal, Joux, Prakash, and Santha proposed an innovative NTRU-like public-key cryptosystem that was believed to be quantum resistant, based on Mersenne prime numbers $q = 2^N - 1$. After a successful attack designed by Beunardeau, Connolly, Géraud, and Naccache, the authors revised the protocol which was accepted for Round 1 of the Post-Quantum Cryptography Standardization Process organized by NIST. The security of this protocol is based on the assumption that a so-called Mersenne Low Hamming Combination Search Problem (MLH-CombSP) is hard to solve. In this work, we present a reduction of MLH-CombSP to Integer Linear Programming (ILP). This opens new research directions for assessing the concrete robustness of such cryptosystem. In particular, we uncover a new family of weak keys, for whose our attack runs in polynomial time.

**Keywords:** Post-Quantum Cryptography, Public-Key Cryptography, Integer Linear Programming, Mersenne-Based Cryptosystem.

## 1 Introduction

In [1], Aggarwal, Joux, Prakash, and Santha introduced a new public-key encryption scheme similar to the NTRU cryptosystem [2] that employs the properties of Mersenne numbers.

A Mersenne number is an integer $q = 2^N - 1$ so that $N$ is prime. One can associate to each element in the ring $\mathbb{Z}_q$ a binary string representing $0 \leq a < q$ of the class $[a] \in \mathbb{Z}_q$. The secret key is a pair of elements $F$ and $G \in \mathbb{Z}_q$ with Hamming weight $h < \sqrt{N/10}$. Let $R$ be chosen at random from $\mathbb{Z}_q$; the public key is given by the pair $(R, T \equiv RF + G \mod q)$. The security assumption (and the mathematical problem that supports the robustness of this cryptosystem) is that it is hard to recover $F$ and $G$, knowing only $R$

and $T$. This assumption is called *Mersenne Low Hamming Combination Search Problem* (MLHCombSP).

The version in [1] is the second iteration of the cryptosystem, first presented in [3]. The security assumptions were based on a problem similar to MLHCombSP and called *Mersenne Low Hamming Ratio Search Problem* (MLHRatioSP). That system has been successfully attacked by Beunardeau et al. in [4]. The attack is performed via a series of calls to an SVP-oracle. Its complexity has been estimated by de Boer et al. in [5]. They also showed that a Meet-in-the-Middle attack is possible using locality-sensitive hashing, which improves upon brute force. However, Beunardeau et al. attack turned out to be the most effective of the two. After the publications of these works, Aggarwal et al. revised the protocol [1] to prevent the above attacks from being effective against the full-scale cipher.

This protocol has been accepted to the Round 1 of the Post-Quantum Cryptography Standardization Process organized by NIST. However, it does not appear among the proposals for Round 2.

## 1.1 Our Contribution/Outline

In this work we present a non-trivial reduction of the underlying mathematical problem of [1] to a relatively low-dimensional Integer Linear Programming (ILP) instance. The secret key is a solution of the resulting ILP instance with probability $p$, that depends on the size of $F$ and $G$.

In section 2, we introduce notation and related work. Furthermore, we recap the Beunardeau et al. attack against [3] with a generalization to the MLHCombSP. Section 3 describes our reduction together with the success probability analysis. There, we describe a variation in the description of the ILP to be solved, that allows some flexibility for the attacker. In particular, one can perform a trade-off between the success probability of the attack and the dimension of the resulting ILP. The application of this trade-off is shown by two examples. In section 4, we describe a new family of weak keys $(F, G)$ and the probability of such a pair to appear. These keys are characterized by a long sequence of zeros in their bit-wise representation. The family is obtained by performing two independent rotations on $F$ and $G$. After these rotations, $F$ and $G$ become small and easy to recover. In this way the size of the set of the weak keys increases. For example, for $N = 1279$ and $h = 17$ (parameters used in [4]), a random key is weak in the sense of Beunardeau et al. with probability $\sim 2^{-34}$. In what follows, we estimate that a random key is weak with probability $\sim 2^{-11}$.

# 2 Preliminaries

**Definition 1.** *Let $N$ be a prime number and let $q = 2^N - 1$. Then $q$ is called a Mersenne number. If $q$ is also prime, then it is called Mersenne prime number.*

Let $\text{seq}_N : \{0, ..., q-1\} \to \{0,1\}^N$ be the map which associates to each $A$ the corresponding $N$-bits binary representation $\text{seq}_N(A)$ with most-significant bit to the left.

Let us consider an integer $0 \leq B < q$, $\text{seq}_N$ maps $[B] \in \mathbb{Z}_q$ to the $N$-bits binary representation of $B$. We define the *Hamming weight* $w(B)$ of $B$ as the Hamming weight of $\text{seq}_N(B)$, i.e. the number of 1s in $\text{seq}_N(B)$.

**Lemma 1.** *Let $k \geq 0$ be a positive integer, let $A$ be an $N$-bits number, and let $q = 2^N - 1$. Then $\text{seq}_N(2^k A \mod q)$ corresponds to a rotation of $\text{seq}_N(A)$ of $k$ positions to the left and $\text{seq}_N(2^{-k} A \mod q)$ corresponds to a rotation of $k$ positions to the right.*

*Proof.* We prove it by induction on $k$. Write $\text{seq}_N(A) = (A_{N-1}, ..., A_1, A_0)$, where $A_{N-1}$ is the most significant bit of $A$. Then we can represent $A$ as

$$A = A_{N-1} \cdot 2^{N-1} + ... + A_1 \cdot 2 + A_0.$$

If we multiply $A$ by 2 modulo $q$ we obtain

$$\begin{aligned} 2 \cdot A &\equiv A_{N-1} \cdot 2^N + A_{N-2} \cdot 2^{N-1} + ... + A_1 \cdot 2^2 + A_0 \cdot 2 \mod q \\ &\equiv A_{N-2} \cdot 2^{N-1} + ... + A_1 \cdot 2^2 + A_0 \cdot 2 + A_{N-1} \mod q. \end{aligned}$$

Then $\text{seq}_N(2 \cdot A) = (A_{N-2}, ..., A_0, A_{N-1})$, i.e. the left rotation of 1 position of $\text{seq}_N(A)$.
By inductive hypothesis, $\text{seq}_N(2^k \cdot A)$ corresponds to the left rotation of $k$ positions of $\text{seq}_N(A)$, then $\text{seq}_N(2^{k+1} \cdot A) = \text{seq}_N(2 \cdot 2^k \cdot A)$ corresponds to the left rotation of one position of $\text{seq}_N(2^k \cdot A)$, that is the left rotation of $k+1$ positions of $\text{seq}_N(A)$. The case right rotations of $\text{seq}_N(A)$ follows trivially. □

The security of the Aggarwal et al. cryptosystem [1] relies on the assumption that the following two problems are hard to solve.

**Mersenne Low Hamming Ratio Search Problem** Let $q = 2^N - 1$ be a Mersenne prime number, $h < N$ an integer, $F$ and $G$ two integers chosen at random from the set

of $N$-bit numbers with Hamming weight $h$. Let $H < q$ be the non-negative integer such that

$$H \equiv \frac{F}{G} \mod q. \tag{2.1}$$

The *Mersenne Low Hamming Ratio Search Problem* (MLHRatioSP) is to find $(F, G)$ given $h$ and $H$.

**Mersenne Low Hamming Combination Search Problem** Let $q = 2^N - 1$ be a Mersenne prime number, $h < N$ an integer, $R$ a random $N$-bit number, and $F, G$ integers chosen at random from the set of $N$-bits numbers with Hamming weight $h$. Let $T < q$ be the non-negative integer such that

$$RF + G \equiv T \mod q. \tag{2.2}$$

The *Mersenne Low Hamming Combination Search Problem* (MLHCombSP) is to find $(F, G)$ given $h$ and the pair $(R, T)$.

In [3], the authors suggest to choose $N$ and $h$ to be such that $\binom{N-1}{h-1} \geq 2^\lambda$ and $4h^2 < N$, for a desired $\lambda$-bit security level. After the publications of the attacks by Beunardeau et al. [4] and De Boer et al. [5], the authors revised the choice of the parameters to be such that $h = \lambda$ and $10h^2 < N$, see [1].

## 2.1 Previous Attacks

**Brute force attack** In [3], Aggarwal et al. showed that a brute force attack to the MLHRatioSP would require $\binom{N-1}{h-1}$ trials. One assumes that one of the two secret numbers, say $F$, has a 1 in the most significant bit (condition that can be obtained by a rotation of $\text{seq}_N(F)$). Then one should check, for every $N$-bits number with 1 as most significant bit and weight $h$, whether the corresponding $G$ through relation (2.1) has weight $h$. This approach does not apply to the MLHCombSP, which instead requires $\binom{N}{h}$ trials.

**Meet-in-the-Middle attack** De Boer et al. [5] showed that a Meet-in-the-Middle attack to MLHRatioSP is possible using locality-sensitive hashing with complexity $\tilde{O}\left(\sqrt{\binom{N-1}{h-1}}\right)$ on classical computers and $\tilde{O}\left(\sqrt[3]{\binom{N-1}{h-1}}\right)$ on quantum computers.

**Weak Keys and Lattice attack** Following the parameters' setting in [3], Beunardeau et al. found a weak key attack to the MLHRatioSP for the case when both $F$ and $G$

happen to have bits set to 1 only in their right halves, i.e. $F, G < \sqrt{2^N}$ [4]. This event happens with probability approximately $2^{-2h}$, for $h << N$.

Following the above idea, Beunardeau et al. also presented a more general attack to the MLHRatioSP which consists of guessing a decomposition of $F$ and $G$ into windows of bits such that all the '1's are "close" to the right-most bit of such windows. Then $F$ and $G$ can be recovered through a lattice reduction algorithm such as LLL [6]. Even if Beunardeau et al. showed that this attack practically hits the security estimations in [3], they did not present any clear asymptotic analysis of its complexity. However, de Boer et al. [5], computed the complexity of this attack.

In [1], the authors stated that the above attack likely generalizes to the MLHCombSP case. Building directly on the work presented in [5], we show in the next subsection that this is true. However we refer the reader to [4] and [5] for a more detailed description.

## 2.2 The Beunardeau et al. attack on MLHCombSP

Since $F$ is taken at random among the $N$-bits numbers with Hamming weight $h$, w.h.p. the '1' valued bits of $\text{seq}_N(F)$ do not appear in big clusters along the $N$ possible positions. One then computes an interval-like partition $\mathcal{P}$ of $\{0, \ldots, N-1\}$ at random, i.e. each set of $\mathcal{P}$ is of the form $\{a, a+1, \ldots, b-1, b\}$, with $0 \leq a < b < N$. Let all '1' valued bits of $\text{seq}_N(F)$ fall in the right-half of one of the sets of $\mathcal{P}$. Then, each set of $\mathcal{P}$ corresponds to a binary substring of $\text{seq}_N(F)$, corresponding in turn to a "small" number. Therefore, the array of these numbers can be seen as a representation of $F$.

Let $\mathcal{P} = \{P_1, .., P_k\}$ and $\mathcal{Q} = \{Q_1, ..., Q_l\}$ be two interval-like partitions of $\{0, ..., N-1\}$ and $(R, T) \in \mathbb{Z}_q^2$ be public parameters of an MLHCombSP instance. Let $p_i, q_i$ be the smallest elements of $P_i, Q_i$ respectively. We consider the following integer lattice.

$$\mathcal{L}_{\mathcal{P},\mathcal{Q},R,T} = \left\{ (x_1, ..., x_k, y_1, ..., y_l, u) \mid R \cdot \sum_{i=1}^{k} 2^{p_i} \cdot x_i + \sum_{j=1}^{l} 2^{q_i} \cdot y_i - uT \equiv 0 \mod q \right\}$$

The above defined lattice $\mathcal{L}_{\mathcal{P},\mathcal{Q},R,T}$ has volume $\det(\mathcal{L}_{\mathcal{P},\mathcal{Q},R,T}) = q$ and rank $d = k + l + 1$. Let $(F, G) \in \mathbb{Z}_q^2$ be such that $w(F) = w(G) = h$ and $RF + G \equiv T$ as in a MLHCombSP instance. Define the vector

$$\boldsymbol{s} = (f_1, ..., f_k, g_1, ..., g_l, 1) \in \mathcal{L}_{\mathcal{P},\mathcal{Q},R,T},$$

where $0 \leq f_i < 2^{|P_i|}$ and $0 \leq g_j < 2^{|Q_j|}$ are the unique natural numbers such that $\sum_{i=1}^{k} f_i \cdot 2^{p_i} = F$ and $\sum_{j=1}^{l} g_j \cdot 2^{q_j} = G$, where $|\cdot|$ denotes the cardinality operator. One wishes to find the vector $\boldsymbol{s}$ by a lattice reduction algorithm applied to $\mathcal{L}_{\mathcal{P},\mathcal{Q},R,T}$.

The lattice $\mathcal{L}_{\mathcal{P},\mathcal{Q},R,T}$ is very similar to the one defined in [5] for the MLHRatioSP and their success probability analysis of the attack holds for this case too. Therefore the following conclusions follow directly from the work of de Boer et al.

Given two partitions $\mathcal{P}$ and $\mathcal{Q}$ of $\{0, ..., N-1\}$ with block size at least $N/d + \Theta(\log N)$, where $d = k + l + 1$ with $k = |P|$ and $l = |Q|$. The success probability of finding the vector $\boldsymbol{s} \in \mathcal{L}_{\mathcal{P},\mathcal{Q},R,T}$ using a SVP-oracle is $2^{-2h+o(1)}$.

**Remark 1.** *The above attack is actually a simplified version of the attack of Beunardeau et al. Indeed, a more general attack can be made by considering the variation of partition sizes and the fraction of each partition block. This variant of the attack has success probability* $2^{-(2+\delta)h+o(1)}$*, for some small constant* $\delta > 0$ *[5].*

**Remark 2.** *In practice, instead of an SVP-oracle, the LLL algorithm [6] which has polynomial complexity is used. This decreases the overall complexity of the attack, but the success probability is decreased too [5].*

The above attack was made against the parameters setting contained in the first version of Aggarwal et al. work. However, as already mentioned, in the most recent version of their work the authors revisited the protocol in order to withstand it.

## 2.3 Integer Linear Programming

An *Integer Linear Programming* (ILP) problem in his *canonical form* is defined as follows. Given a matrix $A \in \mathbb{Q}^{m \times n}$ and two vectors $\boldsymbol{c} \in \mathbb{Q}^n$ and $\boldsymbol{b} \in \mathbb{Q}^m$, minimize (or maximise) the quantity

$$\boldsymbol{c}^T \boldsymbol{x}$$

subject to

$$\begin{cases} A\boldsymbol{x} \leq \boldsymbol{b}, \\ \boldsymbol{x} \geq 0, \\ \boldsymbol{x} \in \mathbb{Z}^n \end{cases}$$

The number $n$ is called the *dimension* of the ILP. An *ILP-oracle* is an oracle that solves any ILP instance.

Solving a general ILP is proved to be NP-hard [7]. Nevertheless, understanding the

complexity of specific families of ILP problems is not an easy task: it can widely vary from case to case [8]. For example, when the problem can be reduced to a simple *Linear Programming* problem, it is proved that it has polynomial complexity [9]. H. Lenstra also provided a polynomial algorithm for certain ILP problems [10].

Nowadays there exists families of ILP solving algorithms, for example *Branch and Bound* [11], *Lagrange relaxation* [12], *Column Generation* [13], and the *Cutting Planes* [14], whose implementations [15, 16] are able to solve in practice relatively challenging instances.

# 3 ILP Reduction

Let $R$, $T$ be two random elements of $\mathbb{Z}_q^*$. We define the map $\varphi : \mathbb{Z}_q \to \mathbb{Z}_q$ sending $X \mapsto -RX + T$. Any point on the graph of $\varphi$, namely $\{(X, \varphi(X))\}_{X \in \mathbb{Z}_q}$, satisfying the condition that both coordinates have Hamming weight equal to $h$ is a solution to the MLHCombSP. We denote such condition as *the graph condition*.

We notice that $\varphi$ is bijective, for it is the combination of two bijective functions (i.e. multiplication times a nonzero element of a field and sum with an element of the underlying group). This means that for any subset $\mathcal{U} \subseteq \mathbb{Z}_q$, the restriction $\varphi_{|\mathcal{U}}$ is injective. Hence, $|\operatorname{Im}(\varphi_{|\mathcal{U}})| = |\mathcal{U}|$.

Let $\mathcal{V}$ be another subset of $\mathbb{Z}_q$, and assume that $\varphi$ behaves like a random bijection from $\mathbb{Z}_q$ to itself. The probability that a random element of $\operatorname{Im}(\varphi_{|\mathcal{U}})$ is in $\mathcal{V}$ is given by $\frac{|\mathcal{V}|}{2^N-1}$. Hence the expected size of $\operatorname{Im}(\varphi_{|\mathcal{U}}) \cap \mathcal{V}$ is given by the mean of the Hypergeometric distribution [17] in $|\mathcal{U}|$ draws, from a population of size $2^N - 1$ that contains $|\mathcal{V}|$ objects that yield a success. That is:

$$\mathbb{E}(|\operatorname{Im}(\varphi_{|\mathcal{U}}) \cap \mathcal{V}|) = \frac{|\mathcal{U}||\mathcal{V}|}{2^N - 1}. \tag{3.1}$$

**Remark 3.** *The expectation* (3.1) *is obtained assuming that $\varphi$ is a random bijection of $\mathbb{Z}_q$. We verified experimentally that this is accurate also for $\varphi(X) = -RX + T$.*

Let $\mathcal{U}, \mathcal{V} \subset \mathbb{Z}_q$ be such that $F \in \mathcal{U}$ and $G \in \mathcal{V}$. Then, $(F, G)$ is a solution of the system of constraints

$$\begin{cases} T - Rx \equiv y \mod q, \\ x \in \mathcal{U}, \\ y \in \mathcal{V}. \end{cases} \tag{3.2}$$

For every fixed instance of $x \in \{0, \ldots, q-1\}$ , there is exactly one $a \in \mathbb{Z}$ that satisfies $0 \leq T + aq - Rx < q$. It is possible to represent (3.2) in terms of integers:

$$\begin{cases} T + qa - Rx = y, \\ x \in \mathcal{U}, \\ y \in \mathcal{V}. \end{cases} \tag{3.3}$$

Assume that $\mathcal{U} = \{\mathrm{l}_{x_3}, \mathrm{l}_{x_3}+1, \ldots, \mathrm{u}_{x_3}-1, \mathrm{u}_{x_3}\}$ and $\mathcal{V} = \{\mathrm{l}_y, \mathrm{l}_y+1, \ldots, \mathrm{u}_y-1, \mathrm{u}_y\}$, for some integers $\mathrm{l}_{x_3}, \mathrm{u}_{x_3}, \mathrm{l}_y, \mathrm{u}_y < q$, and that $F \in \mathcal{U}$ and $G \in \mathcal{V}$. Assume also that (3.3) has a unique solution. Then, one can use an ILP solver to recover $(F, G)$ from the following ILP instance. Minimize the quantity in the integer variables $x_1, x_2, x_3, y$:

$$Tx_1 + qx_2 - Rx_3 + 0y \tag{3.4}$$

with constraints

$$\begin{cases} y = Tx_1 + qx_2 - Rx_3, \\ x_1 = 1, \\ \mathrm{l}_{x_3} \leq x_3 \leq \mathrm{u}_{x_3}, \\ \mathrm{l}_y \leq Tx_1 + qx_2 - Rx_3 \leq \mathrm{u}_y\,. \end{cases} \tag{3.5}$$

Note that $y$ is a redundant variable because it is set to be integer by default ($T, q$ and $R$ are integers) and it takes the value of the minimized quantity. Therefore the ILP (3.4) with constraints (3.5) has dimension 3.

**Remark 4.** *Our approach consists of looking for settings where the ILP has only one solution. For this reason, one can use any linear combination of the variables $x_1, x_2, x_3, y$ in* (3.4) *to define the quantity to minimize.*

Finding good choices on $\mathcal{U}$ and $\mathcal{V}$ (i.e. small and containing $F$ and $G$ with high probability) is, in general, a difficult task. One can exploit the fact that $F$ has weight exactly $h$ to establish the following ILP problem in the integer variables $x_1, x_2, x_3, n_1, \ldots, n_N, y$:

$$Tx_1 + qx_2 - Rx_3 + 0n_1 + \cdots + 0n_N + 0y, \tag{3.6}$$

with constraints

$$\begin{cases} y = Tx_1 + qx_2 - Rx_3, \\ x_1 = 1, \\ x_3 = \sum_{i=1}^{N} n_i 2^{i-1}, \\ 0 \leq n_i \leq 1, \quad \text{for } i = 1, \ldots, N \\ \sum_{i=1}^{N} n_i = h, \\ \mathrm{l}_y \leq y \leq \mathrm{u}_y \,. \end{cases} \tag{3.7}$$

Using these constraints results in having $\mathcal{U}$ of size $|\mathcal{U}| = \binom{N}{h}$. On the other hand, the dimension of the ILP to be solved increased from 3 to $N+3$ (the variable $y$ is redundant). In subsection 3.1, we show how to obtain a trade-off between the number of variables of the ILP to be solved or the size of $\mathcal{U}$.

## 3.1 Merging Bits

To reduce the number of variables of the ILP (3.6), one can merge more than one bit within each variable $n_i$. Consider the $2^s$-ary representation of $x_3 = \sum_{i=1}^{\lceil N/s \rceil} 2^{s(i-1)} n_i$. For $s = 2$, each $n_i$ can assume values in $\{0, 1, 2, 3\}$ and the total weight of $x_3$ varies between $h$ and $2h$, as we prove in Proposition 1. Compared to the setting in (3.6) and (3.7), this results in an increase of the size of $\mathcal{U}$, and decrease of the number of variables: from $N + 3$ to $\lceil N/2 \rceil + 3$.

**Example 1.** *Let $F = (00010011)$ and $h = 3$. By merging bits in pairs and assuming the ILP gives the correct solution, one gets $n_1 = (00)$, $n_2 = (01)$, $n_3 = (00)$, $n_4 = (11)$. The total sum results in $n_1 + n_2 + n_3 + n_4 = 4 \leq 2h = 6$.*

Using this method, it is possible to merge an arbitrary number of bits together. Let $S = \lceil N/s \rceil$. Consider the following system of linear inequalities.

$$\begin{cases} T + aq - Rx = y, \\ 2^h - 1 \leq y \leq 2^N - 2^{N-h}, \\ x = \sum_{i=1}^{S} 2^{s(i-1)} n_i, \\ 0 \leq n_i \leq 2^s - 1, \text{ for } 1 \leq i \leq S, \\ h \leq \sum_{i=1}^{S} n_i \leq 2^{s-1} h. \end{cases} \tag{3.8}$$

We prove in Proposition 1 that a solution $(X, \varphi(X))$ satisfying the graph condition is also a solution to the system of inequalities (3.8) and, therefore, it can be obtained via an ILP-oracle. Generally, choosing larger $s$ implies a decrease of the probability that the

ILP-oracle will return the correct solutions because the number of solutions satisfying the conditions increase.

**Proposition 1.** *Let $F, G \in \mathbb{Z}_q$ such that $\varphi(F) = G$ and the Hamming weight of $\mathrm{seq}_N(F)$ is $h$. Then there exists an integer solution $(x, y, a, n_1, \ldots, n_S)$ that solves the system (3.8), with $x = F$ and $y = G$ .*

*Proof.* The first equation and the first inequality of (3.8) are satisfied by the definition of $\varphi$ and by the fact that $y$ is of weight $h$. The second equation and the second inequality represent the fact that we are writing $x$ in base $2^s$. Hence the only remaining thing to prove is that the last inequality holds.

Let $F = F_0 2^0 + \ldots + F_{N-1} 2^{N-1}$. We notice that $n_i = \sum_{j=0}^{s-1} F_{(i-1)s+j} 2^j$. For the fact that $\sum_{i=0}^{N-1} F(i) = h$, we conclude that

$$\sum_{i=1}^{S} n_i = \sum_{i=1}^{S} \sum_{j=0}^{s-1} F_{(i-1)s+j} 2^j \geq \sum_{i=1}^{S} \sum_{j=0}^{s-1} F_{(i-1)s+j} = h.$$

We prove the second inequality by induction on $h$. For $h = 1$, $n_i$ is a string of weight 1 of $s$ bits. That is at most $2^{s-1}$.

Assuming that the inequality holds for $h-1$. If $n_i \leq 2^{s-1}$ for every $i$, the inequality is satisfied. Hence we assume that there exists one $j$ for which $n_j > 2^{s-1}$. This means that the Hamming weight of $\mathrm{seq}_s(n_j) \geq 2$. Then one gets:

$$\sum_i n_i \leq 2^s + \sum_{i \neq j} n_i.$$

The sum of the Hamming weights of $\mathrm{seq}_s(n_j)$, $j \neq i$ is at most $h - 2$. By inductive hypothesis, it follows that

$$\sum_i n_i \leq 2^s + 2^{s-1}(h-2) = 2^{s-1} h.$$

□

The following Proposition determines the size of $\mathcal{U}$ that one obtains from considering the constraints in (3.8).

**Proposition 2.** *Let $\mathcal{U}$ be the set containing all $0 \leq F < q$, whose $2^s$-ary representation*

*$F = \sum_{i=1}^{S} n_i 2^{i-1}$ satisfies $0 \leq n_i < 2^s$, for $1 \leq i \leq S$ and $h \leq \sum_{i=1}^{S} n_i \leq 2^{s-1}h$. Then*

$$|\mathcal{U}| = \sum_{d=h}^{2^{s-1}h} l_{2^s}(S, d),$$

*where $l_t(n, d)$ is the number of integer solutions to $z_1 + \ldots + z_n = d$, $0 \leq z_i < t$.*

*Proof.* Let $d$ be one of the values of $\sum_{i=1}^{S} n_i$. For each $d$, we consider all the possible configurations of $n_1, \ldots, n_S$. Since each of these is bounded by $2^s - 1$, the number of legitimate configurations is $l_{2^s}(S, d)$. □

## Examples

In Table 3.1 and Table 3.2 we report the dimensions of the ILP instances, for two concrete parameter choices, resulting when varying $s$. For each case, we give the success probability of the attack that is computed as follows. Consider the set

$$\mathcal{V} = \{2^h - 1, 2^h, \ldots, 2^{N-t} - 2^{N-t-h}\},$$

with $t$ satisfying $\log_2(|\mathcal{U}|) \leq t$, and $\mathcal{U}$ constructed as in Proposition 2 so that it contains $F$ by default. Since $\log_2(|\mathcal{V}|) < N - t$, we have that $|\mathcal{U}||\mathcal{V}| < 2^N$. Therefore, because of the estimation in (3.1), if $G \in \mathcal{V}$, then we expect (3.8) to have a unique solution with $x = F$ and $y = G$. In this case, the private key $(F, G)$ can be successfully retrieved with a query to an ILP-oracle and, therefore, the success probability of the attack is the probability that $G \in \mathcal{V}$.

Let $E_G$ be the number of '0' valued bits before the first '1' in $\text{seq}_N(G)$. One wants to compute the probability that, for a fixed $s$, $\log_2(|\mathcal{U}|) \leq E_G$. The random variable $E_G$ is distributed according to the negative hypergeometric distribution [18]:

$$Pr(E_G = t) = \frac{\binom{N-t-1}{N-t-h}}{\binom{N}{N-h}}.$$

In Table 3.1 and Table 3.2 the success probability and the number of ILP variables, computed as $\lceil N/s \rceil + 3$, are presented for a variety of $s$[1]. We notice that the parameters in Table 3.1 violate the guidelines given in [1]. These corresponds to the parameters choice of the first iteration of the protocol [3], and are the ones attacked by Beunardeau et al. [4].

[1]The redundant variable $y$ is not taken into account when computing the number of ILP variables.

| $s$ | Probability of success | Number of variables in ILP |
|---|---|---|
| 1 | $2^{-2.56}$ | 1282 |
| 2 | $2^{-3.97}$ | 643 |
| 3 | $2^{-6.13}$ | 430 |
| 4 | $2^{-9.13}$ | 323 |
| 5 | $2^{-12.94}$ | 259 |
| 6 | $2^{-17.33}$ | 217 |
| 7 | $2^{-21.73}$ | 186 |
| 8 | $2^{-26.07}$ | 163 |
| 9 | $2^{-30.47}$ | 146 |
| 10 | $2^{-34.06}$ | 131 |

Table 3.1: parameters: $N = 1279$ and $h = 17$

| $s$ | Probability of success | Number of variables in ILP |
|---|---|---|
| 1 | $2^{-1.36}$ | 1282 |
| 2 | $2^{-1.78}$ | 643 |
| 3 | $2^{-2.80}$ | 430 |
| 4 | $2^{-4.29}$ | 323 |
| 5 | $2^{-6.26}$ | 259 |
| 6 | $2^{-8.64}$ | 217 |
| 7 | $2^{-11.18}$ | 186 |
| 8 | $2^{-13.71}$ | 163 |
| 9 | $2^{-16.27}$ | 146 |
| 10 | $2^{-18.42}$ | 131 |

Table 3.2: parameters: $N = 1279$ and $h = 11$

**Remark 5.** *It is possible to increase the probability of success by taking into consideration the fact that $G$ has weight $h$ too. In this case, one would need to add more constraints to the set $\mathcal{V}$, as done for the set $\mathcal{U}$, resulting in an increase of the number of ILP variables.*

**Remark 6.** *The approach can be easily adjusted in order to solve the MLHRatioSP by taking $T = 0$.*

# 4 A new family of weak keys

In [4], a family of weak keys was introduced for the MLHRatioSP. Those were the ones for which all the '1' valued bits appeared in the right halves of $\text{seq}_N(F)$ and $\text{seq}_N(G)$. As

noted in [1], one can break keys in this family by performing a rational reconstruction [19] of the quotient $H$ defined by (2.1). A key in this family appears with probability approximately $2^{-2h}$. Many keys which have a long sequence of zeros in the middle of their bit-sequence representation are not considered as weak keys in [4]. However, we show that this is a weakness that can be exploited.

Let $u, v \geq 0$ be two integers. The following transformation

$$(2^{(u-v)}R)(2^v F) + 2^u G \equiv 2^u T \mod q \tag{4.1}$$

gives a new instance of the MLHCombSP where the binary representation of the public values $R$ and $T$ are rotated by $u - v$ and $u$ positions respectively. The secret values $F$ and $G$ are rotated by $v$ and $u$ positions. In practice, the transformation 4.1 allows us to rotate the binary representation of $F$ and $G$ by multiplying $R$ and $T$ times powers of 2. We will say that $F$ is *minimized* by $u$ rotations if

$$2^u F \mod q = \min_{\substack{0 \leq \tilde{u} < N \\ \tilde{u} \in \mathbb{N}}} 2^{\tilde{u}} F \mod q.$$

In particular, the binary representation of $2^u F$ has its longest sequence of consecutive zeros arranged to the left.

Our attack targets keys for which $F$ and $G$ contain long sequences of zeros in their binary representations. The approach consists of *minimizing* through rotations $F$ and $G$ using (4.1), then defining an ILP problem as in (3.4) with exceptionally tight bounds so that the solution is unique.

Assume $F$ and $G$ have been *minimized* already as much as possible through rotations, and let $E_F$ and $E_G$ be respectively the length of the largest sequences of consecutive zeros of $F$ and $G$. In this case, one can set $\mathcal{V} = \{2^{N-E_G-1}, 2^{N-E_G-1}+1, \ldots, 2^{N-E_G}-1\}$ so that $|\mathcal{V}| = 2^{N-E_G-1}$. Let $\mathcal{U} \subset \mathbb{F}_q$ be such that $|\mathcal{U}| < 2^{E_G+1}$ and $F \in \mathcal{U}$. Then, because of estimation (3.1), there is only one expected solution to the system of constraints:

$$\begin{cases} T - Rx \equiv y \mod q, \\ x \in \mathcal{U}, \\ y \in \mathcal{V}. \end{cases} \tag{4.2}$$

and this solution is $x = F$, $y = G$ with high probability. Once found, one can rotate back the solutions to retrieve the original pair. More in general, one expects a unique solution when $E_F + E_G \geq N$, and this can be retrieved by solving an ILP instance with only three variables.

Let $A$ be a random positive $N$-bits integer such that $w(A) = h$, and let $E_A$ be the length of the longest sequence of zeros in its binary representation considering rotations. Let $k \leq N - h$ be a positive integer. Our goal is to evaluate the probability

$$Pr(E_A = k). \tag{4.3}$$

Computing the exact probability (4.3) involves a recursive formula and it is hard in practice. For this reason, we used the following approximation. Consider the set of tuples

$$\Omega_{h,N} = \left\{ (a_1, a_2 \ldots, a_h) | a_1 \geq a_i \geq 0 \text{ for } i > 0, \sum_{i=1}^{h} a_i = N - h \right\}.$$

One can use the elements of $\Omega_{h,N}$ to represents the distribution of zeros in seq$(A)$ after the best rotational shift (the one that minimizes $A$). In particular, $a_1, a_2, ..., a_h$ represents the length of each sequence of consecutive zeros as follows

$$\underbrace{0...0}_{a_1 \ times} \ 1 \ \underbrace{0...0}_{a_2 \ times} \ 1 \ ... \ 1 \ \underbrace{0...0}_{a_h \ times} \ 1.$$

Note that $a_1 \geq a_i$, for $i > 1$, ensures that we are considering already the best shift possible.

**Proposition 3.** *Let $(a_1, a_2, ..., a_h)$ be chosen uniformly at random from $\Omega_{h,N}$. Then*

$$Pr(a_1 = k) = \frac{l_{k+1}(h-1, N-h-k)}{\sum_{i=0}^{N-h} l_{i+1}(h-1, N-h-i)}, \tag{4.4}$$

*where $l_t(n, d)$ is the number of integer solutions to $z_1 + \ldots + z_n = d$, $0 \leq z_i < t$.*

*Proof.* Let $\mathcal{A} : \Omega_{h,N} \to \mathbb{Z}$ be the function that maps $(a_1, a_2, ..., a_h)$ to $a_1$. For $0 \leq k \leq N - h$, $\mathcal{A}^{-1}_{h,N}(k)$ is the subset of $\Omega_{h,N}$ containing the tuples of the form $(k, a_2, \ldots, a_h)$ under the condition that $a_j \leq k$ for every $j$ and that $\sum_{j=2}^{h} a_j = N - h - k$. The number of such tuples is $l_{k+1}(h-1, N-h-k)$. □

We verified experimentally that the above model approximates well (4.3). Proposition 3 allowed us to discover a new family of weak keys, namely, pairs $(F, G)$ such that $E_F + E_G \geq N$. We used (4.4) to derive the following formula

$$Pr(E_F + E_G \geq N) \approx \sum_{a=N}^{2(N-h)} \sum_{k=0}^{a} \frac{l_{k+1}(h-1, N-h-k) l_{a-k+1}(h-1, N-h-k)}{\left( \sum_{i=0}^{N-h} l_{i+1}(h-1, N-h-i) \right)^2}.$$

Note that $E_G$ and $E_F$ are upper bounded by $N-h$, and their lower bound is $N/h-1$ if $h$

divides $N$, $\lfloor N/h \rfloor$ otherwise. For $N = 1279$ and $h = 17$, the expected length of the longest sequence of zeros is approximately 256. For these parameters, there is approximately one key every $2^{11}$ with $E_F + E_G \geq N$, and such keys are weak as explained below. This improves upon Beunardeau et al. work for which 1 over $2^{34}$ keys is weak.

Let $C_{ILP}$ be the cost of solving a system with 3 variables and unique solution as in (4.2). To retrieve the key, one must perform up to $N^2$ rotations to find the one that minimizes $F$ and $G$. Since $E_F$ is unknown, for each rotation, one makes a query to the ILP-oracle for every possible value $E_F = k$ can take. One must try up to $N-h-\lfloor N/h \rfloor+2$ possible $k$ to find a unique solution to (4.2), where $\mathcal{U} = \{2^h - 1, 2^h, \ldots, 2^{N-k} - 2^{N-k-h}\}$ and $\mathcal{V} = \{2^h - 1, 2^h, \ldots, 2^k - 2^{k-h}\}$. The overall cost is $(N - h - \lfloor N/h \rfloor + 2)N^2 C_{ILP} < N^3 C_{ILP}$.

H. Lenstra introduced an algorithm [10] that solves the *decisional ILP problem* with a fixed number of variables in polynomial time in the size of the problem input. This problem consists in deciding whether there exists a solution to the ILP instance satisfying the constraints or not. One can use a *branch and bound*-like approach in combination with Lenstra's algorithm to solve our ILP instances in 3 variables. In particular, one may split $\mathcal{U}$ in two subsets $\mathcal{U}_1$ and $\mathcal{U}_2$, and apply Lenstra's algorithm to decide which one contains the (unique) solution. Assume this is $\mathcal{U}_1$. Then, one applies the same procedure to $\mathcal{U}_1$ and so on. It takes $\log(|\mathcal{U}|)$ steps to isolate and find the solution of the ILP instance. The overall cost $C_{ILP}$ is therefore polynomial in $N$.

# 5 Conclusions and Future Work

This work introduces techniques to reduce the MLHCombSP to ILP. In Section 3, we show how to make a trade-off between the success probability of retrieving the private key via an ILP-oracle query and number of variables of the resulting ILP problem. In general, it is not easy to determine the complexity of an ILP instance. Unlike Linear Programming, the dimension of ILP is not determinant in establishing whether an instance is feasible or not to solve [20]. Therefore the size of the ILPs emerging from the reduction in Section 3 is not necessarily related to their hardness. Unfortunately, the vast majority of the ILP solvers available does not support big numbers arithmetic. This prevented us from performing noteworthy experiments ($N > 60$) on this reduction. With a dedicated implementation, it could be possible to perform such experiments and obtain empirical hints about the complexity of these ILP instances.

In Section 4, we introduce a new family of weak keys for which a key-recovery

attack runs in polynomial time by solving ILP instances with only 3 variables and only one expected solution. In particular, this family is significantly larger than the family of weak keys discovered by Beunardeu et al.

# Bibliography

[1] D. Aggarwal, A. Joux, A. Prakash, and M. Santha, "A New Public-Key Cryptosystem via Mersenne Numbers," in *Advances in Cryptology – CRYPTO 2018*, vol. 10993 of *LNCS*, pp. 459–482, Springer, 2018.

[2] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A Ring-based Public Key Cryptosystem," *Algorithmic Number Theory*, 12 1998.

[3] D. Aggarwal, A. Joux, A. Prakash, and M. Santha, "A New Public-Key Cryptosystem via Mersenne Numbers." Cryptology ePrint Archive, Report 2017/481 , version:20170530.072202, 2017.

[4] M. Beunardeau, A. Connolly, R. Géraud, and D. Naccache, "On the Hardness of the Mersenne Low Hamming Ratio Assumption." Cryptology ePrint Archive, Report 2017/522, 2017.

[5] K. de Boer, L. Ducas, S. Jeffery, and R. de Wolf, "Attacks on the AJPS Mersenne-Based Cryptosystem," in *Post-Quantum Cryptography*, vol. 10786 of *LNCS*, pp. 101–120, Springer, 2018.

[6] A. K. Lenstra, H. W. Lenstra, and L. Lovasz, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[7] C. H. Papadimitriou, "On the Complexity of Integer Programming," in *Journal of ACM*, vol. 28, pp. 765–768, ACM, 1981.

[8] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[9] L. Wolsey, *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization, John Wiley & Sons, 1998.

[10] H. W. Lenstra, *Integer Programming with a Fixed Number of Variables*, vol. 8 of *Mathematics of Operations Research*. Informs, 1983.

[11] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, *Branch-and-Bound Algorithms*, vol. 19 of *Discrete Optimization*. Elsevier Science Publishers, 2016.

[12] M. L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," in *Management Science*, vol. 27, pp. 1–18, Informs, 1981.

[13] L. Appelgren, "A Column Generation Algorithm for a Ship Scheduling Problem," in *Transportation Science*, vol. 3, pp. 53–68, Informs, 02 1969.

[14] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, "Cutting Planes in Integer and Mixed Integer Programming," in *Discrete Applied Mathematics*, vol. 123, pp. 397–446, Elsevier Science Publishers, 2002.

[15] I. CPLEX Optimizer, "IBM ILOG CPLEX Optimization Studio," 2018.

[16] L. Gurobi Optimization, "Gurobi Optimizer Reference Manual," 2018.

[17] G. Casella and R. L. Berger, *Statistical Inference*, vol. 2. Duxbury Pacific Grove, 2002.

[18] K. J. Berry and P. W. Mielke Jr, "The Negative Hypergeometric Probability Distribution: Sampling Without Replacement from a Finite Population," in *Perceptual and Motor Skills*, vol. 86, pp. 207–210, SAGE, 1998.

[19] P. S. Wang, "A P-adic Algorithm for Univariate Partial Fractions," in *Proceedings of the Fourth ACM Symposium on Symbolic and Algebraic Computation*, SYMSAC 1981, pp. 212–217, ACM, 1981.

[20] D. Bertsimas and R. Weismantel, *Optimization Over Integers*. Dynamic Ideas, 2005.

# Paper 2

# Attacks on Integer-RLWE

*Alessandro Budroni, Benjamin Chetioui and Ermes Franch*

**Abstract:** In 2019, Gu Chunsheng introduced Integer-RLWE, a variant of RLWE devoid of some of its efficiency flaws. Most notably, he proposes a setting where one of the security parameters $n$ can be an arbitrary positive integer, contrarily to the typical construction $n = 2^k$. In this paper, we analyze the new problem and implement the classical meet-in-the-middle and lattice-based attacks. We then use the peculiarity of the construction of $n$ to build an improved lattice-based attack in cases where $n$ is composite with an odd divisor. For example, for a certain parameters setting, we reduce the estimated complexity of the attack from $2^{288}$ to $2^{164}$. We also present reproducible experiments confirming our theoretical results.

**Keywords:** Post-quantum cryptography, Meet-in-the-middle, Lattice-based attack, I-RLWE.

## 1 Introduction

In 2006, Regev introduced the Learning With Errors (LWE) [1] problem, one of the most important candidate trapdoors in post-quantum cryptography today. This problem has gained the trust of researchers thanks to its simplicity and its connection to lattice theory, which has been studied for years and provides us with useful security estimates. However, cryptosystems based on LWE present the disadvantage of having large public key sizes. In order to overcome this problem, Lyubashevsky, Peikert and Regev introduced Ring-LWE (RLWE) in 2010 [2], a related problem that allows smaller key sizes and more efficient encryption and decryption.

Let $R = \mathbb{Z}[x]/(x^n + 1)$ and let $R_q = R/qR$, for an integer $n > 1$ and a prime $q$. The *Search* RLWE problem consists in finding the secret $\boldsymbol{s} \in R_q$ given samples of the form $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{as} + \boldsymbol{e}) \in R_q \times R_q$, where $\boldsymbol{e} \in R_q$ is a "small" polynomial drawn from a certain distribution. Another variant of the problem is the *Decision* RLWE, which consists in distinguishing the pairs $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{as}+\boldsymbol{e}) \in R_q \times R_q$ from pairs drawn uniformly at random from $R_q \times R_q$.

However, efficiency varies over different polynomial rings in RLWE and a dedicated

optimization is required for each one of them. To overcome this inconvenience, Gu Chunsheng introduced a variant of RLWE named Integer-RLWE (I-RLWE) [3]. In this new problem the variable $x$ in RLWE is substituted with a prime $q$ and the space of keys $R_q$ is substituted with $\mathbb{Z}_p$, i.e. the set of integers modulo $p = q^n + 1$. The samples are of the form $(a, b = as + e) \in \mathbb{Z}_p \times \mathbb{Z}_p$, where $s = \sum_{i=0}^{n-1} s_i q^i$ and $e = \sum_{i=0}^{n-1} e_i q^i$ such that $s_i$ and $e_i$ are "small".

In his work, Gu also presented a public-key encryption protocol based on I-RLWE. It is therefore important to analyze this problem and gain a better understanding of the security it offers.

It is worth mentioning that a similar work has been done by Aggarwal et al. [4], who introduced an integer-version of the NTRU protocol, and by Beunardeau et al. [5] and de Boer et al. [6], who cryptanalyzed it. Moreover, a module version of I-RLWE is used in ThreeBears [7], a candidate protocol in the NIST Post-Quantum Standardization Process.

## 1.1 Contribution

In this paper, we analyze the complexity of the I-RLWE problem.

We provide some background and notation in Section 2. In Section 3 we adapt two standard attacks to this problem, namely a meet-in-the-middle attack [8] and a lattice-based attack [9]. These two attacks are straightforward to adapt to the problem, thus providing an upper bound for the acceptable complexity of further attacks with minimal effort; studying these attacks is a natural choice. We adapt the meet-in-the-middle attack of Cheon et al. [8] on Decision LWE to Search I-RLWE, and analyze its complexity. Likewise, we produce a lattice-based attack and follow the analysis of Alkim et al. [10] to determine its complexity.

In his work [3], Gu introduces a setting in which $q = 2^t$, instead of a prime, and $n$ can be any positive integer, instead of $n = 2^k$. We exploit this setting to construct a lattice-based attack, similar to Gentry's attack on NTRU-composite [11], for cases where $n$ is neither prime nor a power of two, and $q$ is an arbitrary positive integer. We show in Section 4 how these weak choices of $n$ affect the estimated security of I-RLWE. Furthermore, we provide experiments supporting our theoretical estimates in Section 5. Finally we give our conclusions in Section 6.

# 2 Preliminaries and Notation

We denote the set of the real, rational and integer numbers with $\mathbb{R}, \mathbb{Q}, \mathbb{Z}$ respectively. Bold lower case letters represent vectors. For a given vector $\boldsymbol{v}$, $v_j$ represents its $j$-th component. For a positive integer $p$, we write $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Furthermore, the notation $[a]_p \in \{0, ..., p-1\}$ indicates $a \mod p$ and, similarly, $[\boldsymbol{v}]_p$ is the vector composed by the entries of the integer vector $\boldsymbol{v}$ reduced modulo $p$. The notation $\|\boldsymbol{v}\|$ denotes the Euclidean norm of $\boldsymbol{v}$. Matrices are denoted with upper case bold $\mathbf{M}$.

Let $q$ be an odd prime and let $p = q^n+1$, for $n > 1$ integer. Given $a \in \mathbb{Z}_p \setminus \{p-1\}$, let $a'$ be the integer representative of $a$ in $\{0, ..., p-2\}$. We denote with $\boldsymbol{a} = (a_0, a_1, ..., a_{n-1})$ the vector of its components in base $q$. i.e. $a' = \sum_{i=0}^{n-1} a_i q^i$. Similarly, if we represent $a \neq \frac{p}{2}$ with the integer $a' \in \left\{-\frac{p}{2}+1, ..., \frac{p}{2}-1\right\}$, then we can uniquely write $a' = \sum_{i=0}^{n-1} a_i q^i$, with $a_i \in \left\{-\frac{q-1}{2}, ..., \frac{q-1}{2}\right\}$. Hence we will write $\boldsymbol{a} = (a_0, a_1, ..., a_{n-1}) \in \left\{-\frac{q-1}{2}, ..., \frac{q-1}{2}\right\}^n$. Therefore, through this paper, for every mentioned residue $a \in \mathbb{Z}_p$, we implicitly refer to its corresponding vector of components in base $q$ as $\boldsymbol{a}$. Vice versa, for a given vector of components $\boldsymbol{a}$, we implicitly call $a$ the corresponding residue in $\mathbb{Z}_p$, as explained above.

We use the symbol $\approx_B$ to denote the reflexive and symmetric relation between two vectors $\boldsymbol{x} \approx_B \boldsymbol{y}$ iff $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \leq B$ for some positive integer $B < \frac{q}{2}$. In a natural way we can extend this relation to $x, y \in \mathbb{Z}_p$ applying the relation above to the vectors of the corresponding components in base $q$.

## 2.1 Discrete Gaussian Distributions

In the following we write $x \sim D$ to mean that the random variable $x$ follows the distribution $D$. Let $\rho_{0,\sigma}(x)$ be the probability distribution function of the Gaussian distribution $N(0, \sigma)$ with mean 0 and variance $\sigma^2$. We denote with $D_{\mathbb{Z},\sigma}$ the discrete Gaussian distribution on $\mathbb{Z}$ with mean 0 and variance $\sigma^2$ that assigns to each $a \in \mathbb{Z}$ the probability

$$\frac{\rho_{0,\sigma}(a)}{\sum_{d\in\mathbb{Z}} \rho_{0,\sigma}(d)} = \frac{\exp(-\pi a^2/2\sigma^2)}{\sum_{d\in\mathbb{Z}} \exp(-\pi d^2/2\sigma^2)}.$$

Given $n$ independent random variables $x_1, ..., x_n \sim D_{\mathbb{Z},\sigma}$, we assume $y = \sum_{i=1}^{n} x_i$ follows the distribution $D_{\mathbb{Z},\sigma\sqrt{n}}$. This is a common assumption in this field and it comes from the approximation of the discrete Gaussian distribution with the continuous one. With the notation $\boldsymbol{v} \leftarrow D_{\mathbb{Z}^n,\sigma}$ we indicate a vector in $\mathbb{Z}^n$ with entries sampled independently at random from $D_{\mathbb{Z},\sigma}$.

Furthermore, we denote with $U_{\mathbb{Z}_q}$ the uniform distribution over $\mathbb{Z}_q$ and, similarly, $\boldsymbol{v} \leftarrow U_{\mathbb{Z}_q^n}$ is a vector in $\mathbb{Z}_q^n$ with entries sampled independently and uniformly at random from $\mathbb{Z}_q$.

## 2.2 Lattices

In this subsection we recall some important definitions and notions of lattice theory. For a more detailed resource on this topic, we refer the reader to [12].

A **lattice** is a discrete additive subgroup of $\mathbb{R}^n$. Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_m \in \mathbb{R}^n$ be a set of linearly independent vectors. We define the lattice generated by $\boldsymbol{b}_1, ..., \boldsymbol{b}_m$ as

$$\mathcal{L}(\boldsymbol{b}_1, ..., \boldsymbol{b}_m) = \left\{ \boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{v} = \sum_{i=1}^{m} \alpha_i \boldsymbol{b}_i,\ \alpha_i \in \mathbb{Z} \right\}.$$

A *basis* is any set of linearly independent vectors that generates the lattice as a $\mathbb{Z}$-module and the *dimension* is the number of vectors in a basis. Let $\boldsymbol{B}$ a matrix whose rows form a basis of $\mathcal{L}$, we then define the volume of $\mathcal{L}$ as $\mathrm{Vol}(\mathcal{L}) = \sqrt{\det(\boldsymbol{B}\boldsymbol{B}^T)}$. Unless differently specified, we consider *full-rank* lattices through this paper — that is, the case when $m = n$.

**Definition 1.** *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n \in \mathbb{R}^n$ be a set of linearly independent vectors. We denote with $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*$ the **Gram-Schmidt Orthogonalization** of $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ defined as follows:*

$$\boldsymbol{b}_1^* = \boldsymbol{b}_1, \qquad \boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle}{\|\boldsymbol{b}_j^*\|^2} \boldsymbol{b}_j^*, \quad \text{for } 1 < i \leq n.$$

**Definition 2.** *Given a basis of a lattice $\mathcal{L}$ and a gap factor $\alpha \geq 1$, the **unique Shortest Vector Problem** with a gap factor $\alpha$ ($uSVP_\alpha$) is to find (if it exists) the unique non-zero $\boldsymbol{v} \in \mathcal{L}$ such that any $\boldsymbol{u} \in \mathcal{L}$ with $\|\boldsymbol{u}\| \leq \alpha \|\boldsymbol{v}\|$ is an integral multiple of $\boldsymbol{v}$.*

Estimating the complexity to solve $\mathrm{uSVP}_\alpha$ is a central problem in lattice-based cryptography [9]. The following, known as *Gaussian Heuristic*, gives us an estimate of the length of the shortest vector in a random lattice.

**Heuristic 1.** *Let $\mathcal{L}$ be a full-rank lattice of dimension $n$ and let $\boldsymbol{v} \in \mathcal{L}$ be a shortest non-zero vector. Then*

$$\|\boldsymbol{v}\| \approx \sqrt{\frac{n}{2\pi e}} \cdot \mathrm{Vol}(\mathcal{L})^{1/n}.$$

### 2.3 Integer Ring-Learning With Errors

We give here the definitions for the two versions of the Integer-RLWE (I-RLWE) problem introduced by Gu [3]. Let $q, n$ be two positive integers such that $q$ is prime and $q > n^3$, and let $p = q^n + 1$.

**Definition 3.** *Let $\boldsymbol{s} \leftarrow D_{\mathbb{Z}^n,\sigma}$ be secret and let $s$ be the corresponding element in $\mathbb{Z}_q$. Given an arbitrary number of samples of the form*

$$(a, b = as + e \mod p) \in \mathbb{Z}_p \times \mathbb{Z}_p, \tag{2.1}$$

*where $a \leftarrow U_{\mathbb{Z}_p}$ and $e$ has the vector of components $\boldsymbol{e}$ sampled from $D_{\mathbb{Z}^n,\sigma}$, the **Search Integer-RLWE** problem is to retrieve the secret $s$.*

**Definition 4.** *Let $\boldsymbol{s} \leftarrow D_{\mathbb{Z}^n,\sigma}$ be secret and let $s$ be the corresponding element in $\mathbb{Z}_q$. The **Decision Integer-RLWE** problem is to distinguish with non-negligible advantage between an arbitrary number of samples of the form*

$$(a, b = as + e \mod p) \in \mathbb{Z}_p \times \mathbb{Z}_p, \tag{2.2}$$

*where $a \leftarrow U_{\mathbb{Z}_p}$ and $e$ has the vector of components $\boldsymbol{e}$ sampled from $D_{\mathbb{Z}^n,\sigma}$, and the same number of samples drawn uniformly at random from $\mathbb{Z}_p \times \mathbb{Z}_p$.*

In this paper, we address the Search Integer-RLWE problem, referred in the following sections as I-RLWE. In Section 3, we will consider $n$ to be a power of 2 and $\sigma = \sqrt{n}$, as suggested by Gu [3] in the original definition. However, we will exploit a relaxation on $n$ claimed in Remark 4.1 of [3] to build a more efficient attack in Section 4.

## 3 Standard Attacks

### 3.1 Meet-in-the-Middle attack

A classical meet-in-the-middle (MITM) attack on LWE was previously described [8]. Due to the connection I-RLWE has with LWE, we follow the exact same methodology to perform our attack. We also draw inspiration from the work of de Boer et al. on the AJPS Mersenne-Based Cryptosystem [6].

Consider an I-RLWE sample $(a, b = as + e \mod p)$. Let $v = s \mod q^{n/2}$ and

$y = s - v$. For the MITM approach, we consider the noisy relation

$$ay \approx_B b - av,$$

defined in Section 2.

Let $B$ a positive integer, which parametrizes the probability of finding the right secret depending on $n$. In particular, the probability that a given component of $\boldsymbol{s}$ falls in the range $\{-B, \ldots, B\}$ is given by $P_B = P(x \in \{-B, \ldots, B\} \ : \ x \sim N(0, \sigma))$. It follows that the probability of all the components of $\boldsymbol{s}$ and $\boldsymbol{e}$ to fall in the range $\{-B, \ldots, B\}$ is $P_B^{2n}$.

The MITM attack starts by building the table

$$\mathcal{T} = \left\{(av, v) \ : \ \boldsymbol{v} = (\boldsymbol{x}_1, \boldsymbol{x}_2), \boldsymbol{x}_1 \in \{-B, \ldots, B\}^{n/2}, \boldsymbol{x}_2 \in \{0\}^{n/2}\right\} \subset \mathbb{Z}_p \times \mathbb{Z}_p.$$

The second part of the MITM attack consists in an exhaustive search for $y$ such that $\boldsymbol{y} \in \left\{(\boldsymbol{y}_1, \boldsymbol{y}_2) : \ \boldsymbol{y}_1 \in \{0\}^{n/2}, \boldsymbol{y}_2 \in \{-B, \ldots, B\}^{n/2}\right\}$, and $b - ay \in \mathbb{Z}_p$ is close to $av$, the first component of values in $\mathcal{T}$. If such a case occurs for a given $y$ and a given key-value pair $(av, v) \in \mathcal{T}$, then we set $s' = v + y$, and we compute $e' = b - as' \mod p$. Finally, if we have $e' \approx_B 0$, then $s'$ is a likely candidate for $s$.

The difficult component of this attack lies in determining an efficient search algorithm to find an element in $\mathcal{T}$ that is close to $b - ay \mod p$, as is the case for the same attack on LWE.

We achieve this by applying the Noisy Collision Search described by Cheon et al. [8], with some slight adjustments to fit our problem. As such, the below description is directly adapted from their approach.

**Noisy Collision Search**

In order to efficiently split the search space, Cheon et al. propose a locality sensitive hashing function sgn : $\mathbb{Z}_q \to \{0, 1\}$ defined as $\text{sgn}(x) = 1$, for $x \in \{0, \ldots, \frac{q-1}{2}\}$, and 0 otherwise. Consider the set $V_B = \{-\frac{q-1}{2} + B, \ldots, -B - 1\} \cup \{B, \ldots, \frac{q-1}{2} - B\}$. For any $y$ and $t \in \mathbb{Z}_p$ such that $y \approx_B t$, if $y_i \in V_B$, for a given index $i$, then $\text{sgn}(y_i) = \text{sgn}(t_i)$.

To deal with the case when $y_i \notin V_B$, Cheon et al. define a function sgn' : $\mathbb{Z}_q \to$

$\{0, 1, \times\}$ that returns $\text{sgn}(y)$ if $y \in V_B$, and $\times$ otherwise. $\times$ indicates that the result may be either a 1 or a 0.

### Meet-in-the-Middle Algorithm

Our proposal makes use of two sub-algorithms described in the work of Cheon et al., namely Preprocess and Search [8]. We note that in our case, $m = n$ and otherwise perform slight adjustments so as to fit them to the Search Integer-RLWE. The two algorithms detailed below are thus nearly taken verbatim from the aforementioned paper, where the only changes pertain to the content of $\mathcal{T}$ and $\mathcal{H}$ as well as the accumulation of the results of Search in a list $L$. We define $\text{sgn}(\boldsymbol{x})$ (respectively $\text{sgn'}(\boldsymbol{x})$) to denote the application of sgn (respectively sgn') to each of the components of $\boldsymbol{x}$.

- Preprocess: On input $\mathcal{T} \subset \mathbb{Z}_p \times \mathbb{Z}_p$
  1. Initialize an empty hash table $\mathcal{H}$ with $2^n$ (empty) lists with indexes in $\{0, 1\}^n$.
  2. For each $(t, z) \in \mathcal{T}$,
     - append $(t, z)$ into the list indexed $\text{sgn}(\boldsymbol{t})$.
  3. Return non-empty lists $\mathcal{H}$.

- Search: On input a hash table $\mathcal{H}$, a query $y \in \{x \mid \boldsymbol{x} \in \mathbb{Z}_q^n\}$ and a distance bound $B$,
  1. Initialize an empty list $L$.
  2. For each bin $\in \{0, 1\}^n$ obtained from $\text{sgn'}(\boldsymbol{y})$ by replacing $\times$ by 0 or 1,
     - If $\mathcal{H}$ has a list indexed by bin, for each $(t, z)$ in the list, check whether $\|\boldsymbol{y} - \boldsymbol{t}\|_\infty \leq B$. If so, append $z + y$ to $L$.
  3. Return $L$.

Since our changes do not modify the core of the algorithms, we rely on the proof of correctness provided for the original algorithms.

In the same way, we need to adapt the MITM algorithm provided by Cheon et al.

Pseudocode for this is given by Algorithm 1.

**Algorithm 1:** Meet-in-the-middle attack for Search I-RLWE

**Input:** A sample $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p$
$(n,\ q)$ such that $p = q^n + 1$
a positive integer $B$

**Output:** A list $R$ of candidates for $s$

```
Initialize an empty list R
Compute T = {(av, v) : v = (x_1, x_2), x_1 ∈ {−B, ..., B}^{n/2}, x_2 ∈ {0}^{n/2}}
Run Preprocess on input T to have a hash table H
for y ∈ {x | x ∈ {−B, ..., B}^{n/2}} do
    Concatenate the result of Search on input (H, b − ay, B) to R
end
return R
```

Since both $\boldsymbol{e}$ and $\boldsymbol{s}$ are sampled from the same distribution, we use the same $B$ for the construction of $\mathcal{T}$ and for the Search step of the attack. The choice of $B$ affects both the probability of success and the complexity of the MITM algorithm. A higher accuracy, i.e. larger $B$, gives a better probability of the attack to succeed, but implies also a higher complexity.

### Success of the Attack

We use the empirical three-sigma rule of the normal distribution, also known as the 68-95-99.7 rule, to determine a good value for $B$. Take for example $n = 256$; according to the construction of I-RLWE, we have $\sigma = \sqrt{n} = 16$. The empirical three-sigma rule states that, if we set $B = 3\sigma$, $P_B = P(x \in \{-B, \ldots, B\} \ : \ x \sim N(0, \sigma)) \approx 0.9973$.

In this setting, the probability that $\|(\boldsymbol{s}, \boldsymbol{e})\|_\infty \leq B$ (i.e. that the algorithm succeeds) is about $0.9973^{512} \approx 0.25$. On the other hand, if we set $B = 4\sigma$, then the algorithm will find the right secret with probability about $0.9999^{512} \approx 0.95$.

### Complexity Analysis

According to the construction of $\mathcal{T}$, we write $N_{\mathcal{T}} = |\mathcal{T}| = (2B + 1)^{n/2}$. We assume that the insertion of an element into a list has complexity $O(1)$. Now, for each element in $\mathcal{T}$, Preprocess needs to call sgn $n$ times. It follows that the time cost of Preprocess is $N_{\mathcal{T}} \cdot n$.

Since the size of $V_B$ is $q-4B$, we have that $P(y_i \in V_B) = 1-4B/q$ and $P(y_i \notin V_B) = 4B/q$. The average number of positions with $y_i$ not in $V_B$ is $4nB/q$ and the number of look ups in $\mathcal{T}$ is heuristically $2^{4nB/q}$ (cfr. Lemma 2 and 3 of [8]).

Each one of these lookups returns a list of elements. We are interested in counting the average number of elements contained in one of the lists of $\mathcal{H}$.

**Proposition 1.** *Suppose that for $(t, z) \in \mathcal{T}$, $t$ comes from a uniform distribution over $\mathbb{Z}_q^n$. Then, the average length of a given list in $\mathcal{H}$ is $\frac{N_\mathcal{T}}{2^n}$.*

Search thus finds $O(2^{4nB/q} \cdot \frac{N_\mathcal{T}}{2^n})$ elements. Finally, it must compute $\|\cdot\|_\infty$ for each of them, which has $O(n)$ cost.

We summarize these results in Table 3.1.

Table 3.1: Time cost for noisy search

| Preprocess | Search (per query) |
|---|---|
| $N_\mathcal{T} \cdot n$ | $O(2^{4nB/q} \cdot \frac{N_\mathcal{T}}{2^n} \cdot n)$ |

The full MITM algorithm also consists of two phases. We denote by $T_{\text{pre}}$ the time complexity of the whole preprocessing phase (i.e. the building of $\mathcal{T}$ and the call to Preprocess), and by $T_{\text{search}}$ the time complexity of the whole search phase, and give a cost estimation for them below:

- $T_{\text{pre}}$ consists of $N_\mathcal{T}$ multiplications in $\mathbb{Z}_p$ to build $\mathcal{T}$, added to the cost of executing Preprocess that is $N_\mathcal{T} \cdot n$. Hence, $T_{\text{pre}} = O(N_\mathcal{T})$.
- $T_{\text{search}}$ consists of $N_\mathcal{T}$ queries to Search, thus $T_{\text{search}} = O(N_\mathcal{T} \cdot 2^{4nB/q} \cdot \frac{N_\mathcal{T}}{2^n} \cdot n)$.

## 3.2 Lattice-Based Attack

The most successful approach to solve LWE consists of converting this problem into a lattice problem (e.g. uSVP) and then applying a lattice reduction algorithm that solves it [13]. This approach also provides us with estimates of the security of LWE against lattice attacks based on the complexity of such reduction algorithms. Hence, it is natural to apply a lattice-based attack to I-RLWE.

Consider an I-RLWE sample $(a, b = as + e \mod p)$. One wants to define a lattice that, given a small enough standard deviation $\sigma$, contains the *target* vector $\boldsymbol{v} = (\boldsymbol{s}, \boldsymbol{e}, 1)$

as a shortest vector. Next, one applies a reduction algorithm on a basis of such a lattice in order to find $\boldsymbol{v}$.

Consider the following lattice:

$$\mathcal{L} = \left\{ (\boldsymbol{x}, \boldsymbol{y}, u) \in \mathbb{Z}^n \times \mathbb{Z}^n \times \mathbb{Z} : a \sum_{i=0}^{n-1} x_i q^i + \sum_{j=0}^{n-1} y_j q^j - ub \equiv 0 \mod p \right\}. \tag{3.1}$$

By definition, we have that $\boldsymbol{v} \in \mathcal{L}$. Furthermore, its norm is expected to be $\|\boldsymbol{v}\| \approx \sigma\sqrt{2n}$ because the entries of $\boldsymbol{s}$ and $\boldsymbol{e}$ are distributed according to $D_{\mathbb{Z},\sigma}$. Let us find a basis for $\mathcal{L}$. Define $\boldsymbol{w}^{(i)}$ as the vector formed by the components in base $q$ of $-aq^i \mod p$, for $i = 0, ..., n-1$. We indicate with $\boldsymbol{W}$ the $n \times n$ matrix whose $i$-th row is the $\boldsymbol{w}^{(i)}$ vector. We also define the matrix:

$$\boldsymbol{Q} = \begin{pmatrix} q & -1 & 0 & \dots & 0 & 0 \\ 0 & q & -1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & q & -1 & 0 \\ 0 & 0 & \dots & 0 & q & -1 \\ 1 & 0 & \dots & 0 & 0 & q \end{pmatrix} \in \mathbb{Z}^{n \times n}.$$

Based on the above, we define the following matrix:

$$\boldsymbol{B} = \left(\begin{array}{c|c|c} & & 0 \\ \boldsymbol{I}_n & \boldsymbol{W} & \vdots \\ & & 0 \\ \hline & & 0 \\ \boldsymbol{0}_{n\times n} & \boldsymbol{Q} & \vdots \\ & & 0 \\ \hline 0 \dots 0 & b_0 \dots b_{n-1} & 1 \end{array}\right) \in \mathbb{Z}^{(2n+1)\times(2n+1)},$$

where $b = \sum_{i=0}^{n-1} b_i q^i$. The rows of $\boldsymbol{B}$ form a basis for $\mathcal{L}$ and $\mathrm{Vol}(\mathcal{L}) = |\det(\boldsymbol{B})| = p$.

In the unlikely case of $-aq^i \equiv p-1 \mod p$, for some $0 \leq i \leq n-1$, the basis $\boldsymbol{B}$ is not defined. For this situation, one can exploit the fact that $q$ is coprime with $p$ and use the following general construction to generate a basis for $\mathcal{L}$. Given a lattice $\mathcal{S} = \{(x_0, ..., x_{m-1}) \in \mathbb{Z}^m : \sum_{i=0}^{m-1} x_i c_i \equiv 0 \mod q, c_i \in \mathbb{Z}\}$, where at least one of the $a_i$

is coprime with $p$, one can build a basis for $\mathcal{S}$ as follows. Assume $c_0$ is coprime with $p$ and let $d_i \equiv -c_i/c_0 \mod p$ , for $1 \leq i \leq m-1$. A basis for $\mathcal{S}$ is given by the rows of the following matrix

$$\begin{pmatrix} p & 0 & 0 & \dots & 0 \\ d_1 & 1 & 0 & \dots & 0 \\ d_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ d_{m-1} & 0 & 0 & \dots & 1 \end{pmatrix} \in \mathbb{Z}^{m \times m}.$$

This construction gives in general a basis with larger vectors than the ones of $\boldsymbol{B}$. Therefore, for the cases when $-aq^i \not\equiv p-1 \mod p$ for all $0 \leq i < n$, it is preferable to consider $\boldsymbol{B}$.

### Success Condition and Complexity

The best reduction algorithm known in practice is the Block-Korkine-Zolotarev (BKZ) algorithm [14]. This algorithm a reduced basis by calling an SVP oracle in a smaller dimension $\beta$ a polynomial number of times [15].

By taking the analysis in [10] for the case of LWE as a model, we determine the success condition as follows. The Geometric Series Assumption [16, 13] states that a BKZ-reduced basis of a lattice $\mathcal{L}$ of dimension $d$ is such that

$$\|\boldsymbol{b}_i^*\| = \delta_\beta^{d-2i-1} \cdot \mathrm{Vol}(\mathcal{L})^{1/d}, \quad \text{where } \delta_\beta = \left( (\pi\beta)^{1/\beta} \cdot \frac{\beta}{2\pi e} \right)^{1/2(\beta-1)}.$$

Furthermore, the BKZ algorithm will detect the unique shortest vector $\boldsymbol{v}$ of the lattice if its projection $\pi(\boldsymbol{v})$ onto $\mathrm{Span}\{\boldsymbol{b}_{d-\beta+1}^*, ..., \boldsymbol{b}_d^*\}$ is shorter than the norm of $\boldsymbol{b}_{d-\beta}^*$. Let $\lambda$ be the norm of the such projected vector. Then, the attack will succeed if

$$\lambda \leq \delta_\beta^{2\beta-d-1} \cdot \mathrm{Vol}(\mathcal{L})^{1/d}.$$

In our case, we have that $d = 2n+1$ and $\mathrm{Vol}(\mathcal{L}) = p \approx q^n$. The projection $\pi(\boldsymbol{v})$ of our target vector has expected norm $\approx \sigma\sqrt{\beta}$. We give an heuristic argument for that. Let $\boldsymbol{B}^*$ be a matrix whose rows are $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_d^*$. One can represent $\boldsymbol{v} = (x_1, ..., x_d)\boldsymbol{B}^*$, then $\pi(\boldsymbol{v}) = \sum_{d-\beta<i\leq d} x_i \boldsymbol{b}_i^*$. The norm satisfies

$$\|\pi(\boldsymbol{v})\|^2 = \sum_{d-\beta<i\leq d} x_i^2 \|\boldsymbol{b}_i^*\|^2.$$

The entries of the vector $(x_1\|\boldsymbol{b}_1^*\|, ..., x_d\|\boldsymbol{b}_d^*\|)$ have variance $\sigma$, except for the very last one because $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_d^*$ are orthogonal, and the expectation vector of $\boldsymbol{v}$ is $(0, ..., 0, 1)$. This implies that the entries of $\pi(\boldsymbol{v})$ are distributed according to a shifted $D_{\mathbb{Z},\sigma}$, for a small shift, and $\|\pi(\boldsymbol{v})\| \approx \sigma\sqrt{\beta}$.

In order to succeed with the attack, one must choose $\beta$ to be such that

$$\sigma\sqrt{\beta} \leq \delta_\beta^{2(\beta-n-1)} q^{1/2}. \tag{3.2}$$

Since the complexity of BKZ is mostly ruled by the calls to the SVP oracle in dimension $\beta$, we only take the estimated complexity of this sub-routine into consideration. In the literature, there are two main directions for SVP oracle implementations: lattice sieving and lattice enumeration. Thanks to recent developments [17, 18, 19], lattice sieving took an asymptotic advantage over lattice enumeration. For this reason, we will consider only the estimated complexity provided by lattice sieving, that is $\approx 2^{0.292\beta}$ on classical computers. The memory requirements are of the same order of magnitude.

As in the literature for LWE and RLWE, we use the above estimate to determine the theoretical security of I-RLWE for select parameters.

# 4 Improved Lattice-Based Attack for Weak Choices of $n$

In Remark 4.1 of [3], Gu claims that, for $q = 2^t$, one can choose $n$ to be an arbitrary positive integer, while the usual choice in RLWE-based schemes is of the form $n = 2^k$, for some integer $k$. He justifies this different setting with more efficient encryption and decryption processes in his protocol. In this subsection we introduce a lattice-based attack that exploits the fact that $n$ is neither a prime, nor a power of 2.

Consider the following lemma.

**Lemma 1.** *Let $n \in \mathbb{Z}^+$ such that $n = \hat{n}k$, for some positive integers $\hat{n}$ and $k$, and let $q$ be a positive integer. Then $q^n + 1 \equiv 0 \mod q^{\hat{n}} + 1$ if and only if $k$ is odd.*

*Proof.* Since $n = \hat{n}k$, it follows that $q^n + 1$ as $(q^{\hat{n}})^k + 1$ and $q^{\hat{n}} \equiv -1 \mod q^{\hat{n}} + 1$. It follows that:

$$q^n + 1 \equiv (q^{\hat{n}})^k + 1 \equiv (-1)^k + 1 \equiv 0 \mod q^{\hat{n}} + 1 \Leftrightarrow k \text{ is odd.}$$

□

Let $\hat{n}$ be a divisor of $n$ such that $n/\hat{n}$ is odd. Then $\hat{p} = q^{\hat{n}} + 1$ divides $p = q^n + 1$. Let $x \in \mathbb{Z}_p \setminus \{p-1\}$ be such that the components $(x_0, ..., x_{n-1})$ of its representation in base $q$ follow the distribution $D_{\mathbb{Z},\sigma}$, for a small $\sigma$ with respect to $q$. Then we have that $\hat{x} = (x \mod \hat{p}) \in \mathbb{Z}_{\hat{p}}$ has the following representation in base $q$:

$$\hat{\boldsymbol{x}} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{\hat{n}-1}),$$

where $\hat{x}_i = \sum_{j=0}^{n/\hat{n}-1} (-1)^j x_{j\hat{n}+i}$, for $i = 0, ..., \hat{n}-1$. This can be easily seen by applying the reduction $q^{\hat{n}} \equiv -1 \mod \hat{p}$ to $x = x_0 + x_1 q + x_2 q^2 + ... + x_{n-1} q^{n-1}$. The entries of $\hat{\boldsymbol{x}}$ follow the distribution $D_{\mathbb{Z},\hat{\sigma}}$, where $\hat{\sigma} = \sigma\sqrt{n/\hat{n}}$.

Consider an I-RLWE sample $(a, b = as + e \mod p)$ and let $\hat{a} = a \mod \hat{p}$ and $\hat{b} = b \mod \hat{p}$. Thanks to the Chinese Remainder Theorem, we have that

$$\hat{b} = \hat{a}\hat{s} + \hat{e} \mod \hat{p}, \tag{4.1}$$

where $\hat{s}$ (resp. $\hat{e}$) $= s$ (resp. $e$) $\mod \hat{p}$. In other words, it is possible to obtain a new instance of the I-RLWE problem in a smaller dimension $\hat{n}$ such that we have that $\hat{\boldsymbol{s}}, \hat{\boldsymbol{e}} \sim D_{\mathbb{Z}^{\hat{n}},\hat{\sigma}}$, where $\hat{\sigma} = \sigma\sqrt{n/\hat{n}}$.

The attack introduced in this section can be divided in two steps. First, one performes the lattice attack explained in Subsection 3.2 on the I-RLWE problem (4.1) to retrieve $\hat{s}$ and $\hat{e}$. Then, one performs a second lattice reduction to finally retrieve $s$ and $e$ from a lattice with larger volume.

Consider the following lattice:

$$\mathcal{L}_1 = \left\{ (\boldsymbol{x}, \boldsymbol{y}, u) \in \mathbb{Z}^{\hat{n}} \times \mathbb{Z}^{\hat{n}} \times \mathbb{Z} : \hat{a} \sum_{i=0}^{\hat{n}-1} x_i q^i + \sum_{j=0}^{\hat{n}-1} y_j q^j - u\hat{b} \equiv 0 \mod \hat{p} \right\}. \tag{4.2}$$

Analogously to the lattice defined in Subsection 3.2, $\mathcal{L}_1$ contains the *reduced* target vector $\hat{\boldsymbol{v}} = (\hat{\boldsymbol{s}}, \hat{\boldsymbol{e}}, 1)$ and its volume is $\mathrm{Vol}(\mathcal{L}_1) = \hat{p} \approx q^{\hat{n}}$. One can apply a lattice reduction algorithm to find $\hat{\boldsymbol{v}}$ and so the reduced secret $\hat{s}$ and error $\hat{e}$. Next, we define the following lattice:

$$\mathcal{L}_2 = \left\{ (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}) \in \mathbb{Z}^n \times \mathbb{Z}^n \times \mathbb{Z}^3 : \begin{array}{c} x - u_1\hat{s} \equiv 0 \mod \hat{p}, \\ y - u_2\hat{e} \equiv 0 \mod \hat{p}, \\ a\sum_{i=0}^{n-1} x_i q^i + \sum_{j=0}^{n-1} y_j q^j - u_3 b \equiv 0 \mod p \end{array} \right\}, \tag{4.3}$$

where $\boldsymbol{u} = (u_1, u_2, u_3)$. This lattice contains the target vector $\boldsymbol{v} = (\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{1})$, where

$\mathbf{1} = (1, 1, 1)$, and, as there are more conditions on its vectors, we expect it to have a larger volume and dimension compared to the lattice defined by (3.1).

The approach we followed to write a basis for $\mathcal{L}_2$ varies according to the relations between $\text{GCD}(b, p)$, $\text{GCD}(\hat{s}, \hat{p})$ and $\text{GCD}(\hat{e}, \hat{p})$. For sake of conciseness, we show how to build a basis for the attacker's best case scenario, i.e. when $\text{GCD}(b, p) = \text{GCD}(\hat{s}, \hat{p}) = \text{GCD}(\hat{e}, \hat{p}) = 1$.

Consider the following matrix:

$$\boldsymbol{B}_2 = \left(\begin{array}{cc|ccc} & & v_1 & 0 & y_1 \\ \boldsymbol{I}_n & \mathbf{0}_{n\times n} & \vdots & \vdots & \vdots \\ & & v_n & 0 & y_n \\ \hline & & 0 & w_1 & y_{n+1} \\ \mathbf{0}_{n\times n} & \boldsymbol{I}_n & \vdots & \vdots & \vdots \\ & & 0 & w_n & y_{2n} \\ \hline 0\dots0 & 0\dots0 & \hat{p} & 0 & 0 \\ 0\dots0 & 0\dots0 & 0 & \hat{p} & 0 \\ 0\dots0 & 0\dots0 & 0 & 0 & p \end{array}\right) \in \mathbb{Z}^{(2n+3)\times(2n+3)},$$

where

$$v_i = q^{i-1}\hat{s}^{-1} \mod \hat{p} \qquad i = 1, ..., n,$$

$$w_i = q^{i-1}\hat{e}^{-1} \mod \hat{p} \qquad i = 1, ..., n,$$

$$y_i = \begin{cases} aq^{i-1}b^{-1} \mod p & \text{if } \ i = 1, ..., n, \\ q^{i-1}b^{-1} \mod p & \text{if } \ i = n+1, ..., 2n. \end{cases}$$

It is easy to check that $B_2$ is a basis of $\mathcal{L}_2$ and $\text{Vol}(\mathcal{L}_2) = p\hat{p}^2$. In general, $\text{Vol}(\mathcal{L}_2)$ is upper bounded by $p\hat{p}^2 \approx q^{n+2\hat{n}}$. This bound is reached in the attacker's best case scenario.

## 4.1 Analysis and Success Condition

In order for the attack to be successful, the reduced vector $\hat{\boldsymbol{v}} = (\hat{\boldsymbol{s}}, \hat{\boldsymbol{e}}, 1)$ must be small enough to be a shortest vector of $\mathcal{L}_1$. The entries of this vector (except the last one that is set to 1) follow the distribution $D_{\mathbb{Z},\hat{\sigma}}$, where $\hat{\sigma} = \sigma\sqrt{n/\hat{n}}$. Therefore, its expected norm is $\approx \sigma\sqrt{2\hat{n}+1}$. Using the Gaussian Heuristic, we check whether $\hat{\boldsymbol{v}}$ is shorter than the

estimated shortest vector in $\mathcal{L}_1$:

$$\|\hat{\boldsymbol{v}}\| \approx \hat{\sigma}\sqrt{2\hat{n}+1} = \sigma\sqrt{\frac{n}{\hat{n}}}\sqrt{2\hat{n}+1} \leq \sqrt{\frac{2\hat{n}+1}{2\pi e}} \cdot q^{1/2}.$$

Then, one gets that $\sigma$ must be such that:

$$\sigma \leq \sqrt{q\frac{\hat{n}}{2n\pi e}}. \tag{4.4}$$

In his paper, Gu suggested $\sigma = \sqrt{n}$ and $q > n^3$. In this setting, condition (4.4) is satisfied.

We give a condition on the block size $\beta_1$ for the BKZ-$\beta_1$ reduction algorithm to find the target vector $\hat{\boldsymbol{v}}$, using an approach as in Subsection 3.2. For a lattice of dimension $2\hat{n}+1$ and volume $\hat{p}$, $\beta_1$ must be such that:

$$\hat{\sigma}\sqrt{\beta_1} \leq \delta_{\beta_1}^{2(\beta_1-\hat{n}-1)} \cdot q^{\frac{\hat{n}}{2\hat{n}+1}}.$$

Similarly, the target vector $\boldsymbol{v}$ will be found through a BKZ-$\beta_2$ reduction on a basis of $\mathcal{L}_2$ if the block size $\beta_2$ is such that

$$\sigma\sqrt{\beta_2} \leq \delta_{\beta_2}^{2(\beta_2-n-2)} \cdot q^{\frac{n+2\hat{n}}{2n+3}},$$

where the dimension of the lattice is $2n+3$ and the volume is $\mathrm{Vol}(\mathcal{L}_2) = p\hat{p}^2 \approx q^{n+2\hat{n}}$.

In Table 4.1 we show the significant advantage of using this approach over the standard lattice attack described in Subsection 3.2 for some choices of $n$ and $\hat{n}$. The complexity, based on the required cost for performing lattice sieving, drops significantly. This allows us to conclude that $n$ must not have odd divisors, that is to say $n$ is either a prime or a power of 2, in line with the setting of RLWE.

| Parameters | | | Standard Lattice Attack | | Improved Lattice Attack | | |
|---|---|---|---|---|---|---|---|
| $n$ | $\hat{n}$ | $q$ | $\beta$ | Complexity | $\beta_1$ | $\beta_2$ | Complexity |
| 2000 | 400 | $2^{33}$ | 987 | 288 | 130 | 561 | 164 |
| 1500 | 300 | $2^{32}$ | 713 | 208 | 83 | 396 | 116 |
| 1200 | 240 | $2^{31}$ | 559 | 163 | $< 60$ | 304 | 89 |
| 1000 | 200 | $2^{30}$ | 463 | 135 | $< 60$ | 246 | 71 |

Table 4.1: Columns 1, 2 and 3 define the parameters, with $\sigma = \sqrt{n}$. Columns 4 and 7 contain the minimum block size ($\beta$ and $\beta_2$) of the BKZ subroutine required to find the target vector $\boldsymbol{v}$ respectively from lattice (3.1) and (4.3). Column 6 contains the minimum block size $\beta_1$ to find $\hat{\boldsymbol{v}}$ from reducing a basis of lattice (4.2). The complexities in column 5 and 8 are expressed in $\log_2$ and correspond to the lattice sieving complexity with parameter respectively $\beta$ and $\beta_2$

**Remark 1.** *This attack can be further improved when n has more than one odd divisor by adding more conditions in the definition of* $\mathcal{L}_2$.

**Remark 2.** *We remark that these choices of n remain weak for any q and not only in the setting that Gu proposes.*

# 5 Experiments

In order to confirm our theoretical results, we performed some practical experiments which we report in this section.

First we generated some I-RLWE samples, then we used the BKZ implementation contained in the General Sieve Kernel [20], the cutting-edge implementation at the moment of writing, in order to perform the attacks. Finally we compared the minimum block size parameter $\beta$ of the BKZ reduction required to successfully retrieve the secret and the error for both approaches. For each instance, we empirically chose $\hat{n}$ among the possible choices so that the uSVP on the corresponding $\mathcal{L}_1$ is the easiest. In particular, the uSVP on $\mathcal{L}_1$ for our parameters choice was solvable using BKZ with $\beta = 1$, i.e., with the LLL algorithm[1].

[1]Note that the LLL algorithm does not guarantee to solve uSVP in general for dimensions $> 2$.

In the table below we report the results obtained during our experiments. The I-RLWE samples that we used in our experiments can be found at https://archive.org/details/irlwesamples.

| Parameters | | | Standard Lattice Attack | Improved Lattice Attack | |
|---|---|---|---|---|---|
| $n$ | $\hat{n}$ | $q$ | $\beta$ | $\beta_1$ | $\beta_2$ |
| 130 | 26 | $2^{22}$ | 41 | 1 | 2 |
| 110 | 22 | $2^{21}$ | 28 | 1 | 2 |
| 105 | 15 | $2^{21}$ | 9 | 1 | 2 |

Table 5.1: Columns 1, 2 and 3 define the parameters, with $\sigma = \sqrt{n}$. Column 3 report the minimum block size $\beta$ that allowed us to retrieve the target vector $\boldsymbol{v}$ through BKZ reduction on the lattice defined in (3.1). Similarly, columns 4 and 5 report the minimum block sizes $\beta_1$ and $\beta_2$ for lattices (4.2) and (4.3) respectively, so that the attack was successful. Note that BKZ with $\beta_1 = 1$ corresponds to LLL.

# 6 Conclusion

In this work, we adapted a meet-in-the-middle attack and a lattice-based attack from LWE to I-RLWE. The latter, as in the case of LWE and RLWE, gives us theoretical estimates regarding the security provided by I-RLWE.

We introduced a new lattice-based attack against I-RLWE when the parameter $n$ is chosen as a composite number divisible by an odd number. This attack exploits the weakness on the choice of $n$ to build a new lattice of larger volume, leading to a more efficient secret and error recovery through lattice reduction. We provided theoretical estimates of our attack showing how the complexity of solving I-RLWE reduces in this setting. For example, for $n = 2000$ the complexity reduces from $2^{288}$, estimated with the standard lattice attack, to $2^{164}$. This gap also appears for smaller $n$ as in the case for $n = 1000$ where the complexity drops from $2^{135}$ to $2^{71}$. This attack likely applies to RLWE; however, this was not investigated as the setting considered here is avoided in the literature of RLWE-based protocols.

To confirm our theoretical results, we run experiments for $n$ up to 130. Our results shows that a much smaller block-size parameter $\beta$ is required in the BKZ lattice reduction

algorithm in order to successfully recover the secret and the error.

We conclude remarking that choices of $n$ as in the aforementioned case must definitely be avoided in I-RLWE, as is prescribed for RLWE.

# Bibliography

[1] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pp. 84–93, ACM, 2005.

[2] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Advances in Cryptology – EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 1–23, Springer, 2010.

[3] C. Gu, "Integer Version of Ring-LWE and Its Applications," in *Security and Privacy in Social Networks and Big Data - 5th International Symposium, SocialSec 2019, Copenhagen, Denmark*, vol. 1095 of *Communications in Computer and Information Science*, pp. 110–122, Springer, 2019.

[4] D. Aggarwal, A. Joux, A. Prakash, and M. Santha, "A New Public-Key Cryptosystem via Mersenne Numbers," in *Advances in Cryptology – CRYPTO 2018*, vol. 10993 of *LNCS*, pp. 459–482, Springer, 2018.

[5] M. Beunardeau, A. Connolly, R. Géraud, and D. Naccache, "On the Hardness of the Mersenne Low Hamming Ratio Assumption," in *Progress in Cryptology – LATINCRYPT 2017*, vol. 11368 of *LNCS*, pp. 166–174, Springer, 2019.

[6] K. de Boer, L. Ducas, S. Jeffery, and R. de Wolf, "Attacks on the AJPS Mersenne-Based Cryptosystem," in *Post-Quantum Cryptography*, vol. 10786 of *LNCS*, pp. 101–120, Springer, 2018.

[7] M. Hamburg, "Threebears." Technical report, National Institute of Standards and Technology, 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions.

[8] J. H. Cheon, M. Hhan, S. Hong, and Y. Son, "A Hybrid of Dual and Meet-in-the-Middle Attack on Sparse and Ternary Secret LWE," in *IEEE Access*, vol. 7, pp. 89497–89506, IEEE, 2019.

[9] M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, "On the Efficacy of Solving LWE by Reduction to Unique-SVP," in *Information Security and Cryptology – ICISC 2013*, vol. 8565 of *LNCS*, pp. 293–310, Springer, 2014.

[10] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-Quantum Key Exchange: A New Hope," in *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, p. 327–343, USENIX, 2016.

[11] C. Gentry, "Key Recovery and Message Attacks on NTRU-Composite," in *Advances in Cryptology — EUROCRYPT 2001*, vol. 2045 of *LNCS*, pp. 182–194, Springer, 2001.

[12] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 2nd ed., 2014.

[13] M. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the Expected Cost of Solving uSVP and Applications to LWE," in *Advances in Cryptology - ASIACRYPT 2017*, vol. 10625 of *LNCS*, pp. 297–322, Springer, 2017.

[14] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better Lattice Security Estimates," in *Advances in Cryptology – ASIACRYPT 2011*, vol. 7073 of *LNCS*, pp. 1–20, Springer, 2011.

[15] G. Hanrot, X. Pujol, and D. Stehlé, "Terminating BKZ." Cryptology ePrint Archive, Report 2011/198, 2011. https://eprint.iacr.org/2011/198.

[16] Y. Chen, "Lattice Reduction and Concrete Security of Fully Homomorphic Encryption." PhD Thesis, l'Université Paris Diderot, 2013. https://archive.org/details/PhDChen13.

[17] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, p. 10–24, SIAM, 2016.

[18] L. Ducas, "Shortest Vector from Lattice Sieving: A Few Dimensions for Free," in *Advances in Cryptology – EUROCRYPT 2018* (J. B. Nielsen and V. Rijmen, eds.), vol. 10820 of *LNCS*, pp. 125–145, Springer, 2018.

[19] T. Laarhoven and A. Mariano, "Progressive Lattice Sieving," in *Post-Quantum Cryptography*, vol. 10786 of *LNCS*, pp. 292–311, Springer, 2018.

[20] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in

*Advances in Cryptology – EUROCRYPT 2019*, vol. 11477 of *LNCS*, pp. 717–746, Springer, 2019.

# Paper 3

# Improvements on Making BKW Practical for Solving LWE

*Alessandro Budroni, Qian Guo, Thomas Johansson, Erik Mårtensson and Paul Stankovski Wagner*

**Abstract:** The Learning with Errors (LWE) problem is one of the main mathematical foundations of post-quantum cryptography. One of the main groups of algorithms for solving LWE is the Blum-Kalai-Wasserman (BKW) algorithm. This paper presents new improvements of BKW-style algorithms for solving LWE instances. We target minimum concrete complexity and we introduce a new reduction step where we partially reduce the last position in an iteration and finish the reduction in the next iteration, allowing non-integer step sizes. We also introduce a new procedure in the secret recovery by mapping the problem to binary problems and applying the Fast Walsh Hadamard Transform. The complexity of the resulting algorithm compares favorably to all other previous approaches, including lattice sieving. We additionally show the steps of implementing the approach for large LWE problem instances. We provide two implementations of the algorithm, one RAM-based approach that is optimized for speed and one file-based approach which overcomes RAM limitations by using file-based storage

**Keywords:** BKW, LWE, Lattice-Based Cryptography, FWHT, Post-Quantum Cryptography.

## 1 Introduction

Since a large-scale quantum computer easily breaks both the problem of integer factoring and the discrete logarithm problem [1], public-key cryptography needs to be based on other underlying mathematical problems. In post-quantum cryptography - the research area studying such replacements - lattice-based problems are the most promising candidates. In the NIST post-quantum standardization competition, 5 out of 7 finalists and 2 out of 8 alternates are lattice-based [2].

The Learning with Errors problem (LWE) introduced by Regev in [3], is the main

problem in lattice-based cryptography. It has a theoretically very interesting average-case to worst-case reduction to standard lattice-based problems. It has many cryptographic applications, including but not limited to, design of Fully Homomorphic Encryption Schemes (FHE). An interesting special case of LWE is the Learning Parity with Noise problem (LPN), introduced in [4], which has interesting applications in light-weight cryptography.

Considerable cryptanalytic effort has been spent on algorithms for solving LWE. These can be divided into three categories: lattice-reduction, algebraic methods and combinatorial methods. The algebraic methods were introduced by Arora and Ge in [5] and further considered in [6]. For very small noise these methods perform very well, but otherwise the approach is inefficient. The methods based on lattice-reduction are currently the most efficient ones in practise. One way of comparing the different approaches is through the Darmstadt LWE Challenges [7], where the lattice-based approach called General Sieve Kernel (G6K) is the currently most successful algorithm in breaking challenges [8]. The combinatorial algorithms are all based on the Blum-Kalai-Wasserman (BKW) algorithm, and these will be the focus of this paper.

For surveys on the concrete and asymptotic complexity of solving LWE, see [9] and [10, 11], respectively. In essence, BKW-style algorithms have a better asymptotic performance than lattice-based approaches for parameter choices with large noise. Unlike lattice-based approaches, BKW-style algorithms pay a penalty when the number of samples is limited (like in the Darmstadt challenges and many cryptographic schemes). A recent example of a scheme allowing for a very large number of LWE samples is found in [12].

## 1.1 Related Work

The BKW algorithm was originally developed as the first subexponential algorithm for solving the LPN problem [13]. In [14] the algorithm was improved, introducing new concepts like LF2 and the use of the fast Walsh-Hadamard transform (FWHT) for the distinguishing phase. A new distinguisher using subspace hypothesis testing was introduced in [15, 16].

The BKW algorithm was first applied to the LWE problem in [17]. This idea was improved in [18], where the idea of Lazy Modulus Switching (LMS) was introduced. The idea was improved in [19, 20], where [19] introduced so called coded-BKW steps. The idea of combining coded-BKW or LMS with techniques from lattice sieving [21] lead to the next improvement [22]. This combined approach was slightly improved in [11, 23].

The distinguishing part of the BKW algorithm for solving LWE was improved by using the Fast Fourier Transform (FFT) in [24]. One drawback of BKW is its high memory-usage. To remedy this, time-memory trade-offs for the BKW algorithm were recently studied in [25, 26, 27]. A recent fast implementation of the BKW algorithm for solving LPN is in [28].

## 1.2 Contributions

In this paper we introduce a new BKW-style algorithm including the following.

- A generalized reduction step that we refer to as smooth-LMS, allowing us to use non-integer step sizes. These steps allow us to use the same time, space and sample complexity in each reduction step of the algorithm, which improves performance compared to previous work.
- A binary-oriented method for the guessing phase, transforming the LWE problem into an LPN problem. While the previous FFT method guesses a few positions of the secret vector and finds the correct one, this approach instead finds the least significant bits of a large amount of positions using the FWHT. This method allows us to correctly distinguish the secret with a larger noise level, generally leading to an improved performance compared to the FFT based method. In addition, the FWHT is much faster in implementation.
- Concrete complexity calculations for the proposed algorithm showing the lowest known complexity for some parameter choices selected as in the Darmstadt LWE Challenge instances, but with unrestricted number of samples.
- Two implementations of the algorithm that follow two different strategies in memory-management. One is fast, light, and uses solely RAM-memory. The latter follows a file-based strategy to overcome the memory limitations imposed by using only RAM. The file read/write is minimized by implementing the algorithm in a clever way. Simulation results on solving larger instances are presented and verifies the previous theoretical arguments.

## 1.3 Organization

We organize the rest of the paper as follows. We introduce some necessary background in Section 2. In Section 3 we cover previous work on applying the BKW algorithm to the

LWE problem. Then in Section 4 we introduce our new Smooth-LMS reduction method. Next, in Section 5 we go over our new binary-oriented guessing procedure. Sections 6 and 7 cover the complexity analysis and implementation of our algorithm, respectively. Section 8 describes our experimental results using the implementation. Finally, the paper is concluded in Section 9.

# 2 Background

## 2.1 Notation

Throughout the paper we use the following notations.

- We write $\log(\cdot)$ for the base 2 logarithm.
- In the $n$-dimensional Euclidean space $\mathbb{R}^n$, by the norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ we consider its $L_2$-norm, defined as

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}.$$

  The Euclidean distance between vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ is defined as $\|\mathbf{x} - \mathbf{y}\|$.
- Elements in $\mathbb{Z}_q$ are represented by the set of integers in $[-\frac{q-1}{2}, \frac{q-1}{2}]$.
- For an $[N, k]$ linear code, $N$ denotes the code length and $k$ denotes the dimension.

## 2.2 The LWE and LPN Problems

The LWE problem [3] is defined as follows.

**Definition 1.** *Let $n$ be a positive integer, $q$ a prime, and let $\mathcal{X}$ be an error distribution selected as the discrete Gaussian distribution on $\mathbb{Z}_q$ with variance $\sigma^2$. Fix $\mathbf{s}$ to be a secret vector in $\mathbb{Z}_q^n$, chosen from some distribution (usually the uniform distribution). Denote by $L_{\mathbf{s},\mathcal{X}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error $e \in \mathbb{Z}_q$ from $\mathcal{X}$ and returning*

$$(\mathbf{a}, z) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

*in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The (search)* LWE *problem is to find the secret vector $\mathbf{s}$ given a fixed number of samples from $L_{\mathbf{s},\mathcal{X}}$.*

The definition above gives the *search* LWE problem, as the problem description asks for the recovery of the secret vector $\mathbf{s}$. Another version is the *decision* LWE problem, in which case the problem is to distinguish between samples drawn from $L_{\mathbf{s},\mathcal{X}}$ and a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Let us also define the LPN problem, which is a binary special case of LWE.

**Definition 2.** *Let $k$ be a positive integer, let $\mathbf{x}$ be a secret binary vector of length $k$ and let $X \sim \mathsf{Ber}_\eta$ be a Bernoulli distributed error with parameter $\eta > 0$. Let $L_{\mathbf{x},X}$ denote the probability distribution on $\mathbb{F}_2^k \times \mathbb{F}_2$ obtained by choosing $\mathbf{g}$ uniformly at random, choosing $e \in \mathbf{F}_2$ from $X$ and returning*

$$(\mathbf{g}, z) = (\mathbf{g}, \langle \mathbf{g}, \mathbf{x} \rangle + e)$$

*The (search)* LPN *problem is to find the secret vector $\mathbf{x}$ given a fixed number of samples from $L_{\mathbf{x},X}$.*

Just like for LWE, we can also, analogously, define *decision* LPN.

Previously, analysis of algorithms solving the LWE problem have used two different approaches. One being calculating the number of operations needed to solve a certain instance for a particular algorithm, and then comparing the different complexity results. The other being asymptotic analysis. Solvers for the LWE problem with suitable parameters are expected to have fully exponential complexity, bounded by $2^{cn}$ as $n$ tends to infinity, where the value of $c$ depends on the algorithms and the parameters of the involved distributions. In this paper, we focus on the complexity computed as the number of arithmetic operations in $\mathbb{Z}_q$, for solving particular LWE instances (and we do not consider the asymptotics).

## 2.3 Discrete Gaussian Distributions

We define the discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and variance $\sigma^2$, denoted $D_{\mathbb{Z},\sigma}$ as the probability distribution obtained by assigning a probability proportional to $\exp(-x^2/(2\sigma^2))$ to each $x \in \mathbb{Z}$. Then, the discrete Gaussian distribution $\mathcal{X}$ over $\mathbb{Z}_q$ with variance $\sigma^2$ (also denoted $\mathcal{X}_\sigma$) can be defined by folding $D_{\mathbb{Z},\sigma}$ and accumulating the value of the probability mass function over all integers in each residue class modulo $q$. It makes sense to consider the noise level as $\alpha$, where $\sigma = \alpha q$. We also define the rounded Gaussian distribution on $Z_q$. This distribution samples values by sampling values from the continuous Gaussian distribution with mean 0 and variance $\sigma^2$, rounding to the

closest integer and then folding the result to the corresponding value in $\mathbb{Z}_q$. We denote it by $\bar{\Psi}_{\sigma,q}$.

If two independent $X_1$ and $X_2$ are drawn from $\mathcal{X}_{\sigma_1}$ and $\mathcal{X}_{\sigma_2}$ respectively, we make the heuristic assumption that their sum is drawn from $\mathcal{X}_{\sqrt{\sigma_1^2+\sigma_2^2}}$. We make the corresponding assumption for the rounded Gaussian distribution.

# 3 A Review of BKW-style Algorithms

## 3.1 The LWE Problem Reformulated

Assume that $m$ samples

$$(\mathbf{a}_1, z_1), (\mathbf{a}_2, z_2), \ldots, (\mathbf{a}_m, z_m),$$

are collected from the LWE distribution $L_{\mathbf{s},\mathcal{X}}$, where $\mathbf{a}_i \in \mathbb{Z}_q^n, z_i \in \mathbb{Z}_q$. Let $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m) = \mathbf{sA}$. We have

$$\mathbf{z} = \mathbf{sA} + \mathbf{e},$$

where $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} & \mathbf{a}_2^{\mathrm{T}} & \cdots & \mathbf{a}_m^{\mathrm{T}} \end{bmatrix}$, $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$ and $e_i \xleftarrow{\$} \mathcal{X}$. The search LWE problem is a decoding problem, where $\mathbf{A}$ serves as the generator matrix for a linear code over $\mathbb{Z}_q$ and $\mathbf{z}$ is a received word. Finding the secret vector $\mathbf{s}$ is equivalent to finding the codeword $\mathbf{y} = \mathbf{sA}$ for which the Euclidean distance $||\mathbf{y} - \mathbf{z}||$ is minimal. In the sequel, we adopt the notation $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$.

## 3.2 Transforming the Secret Distribution

A transformation [29, 30] can be applied to ensure that the secret vector follows the same distribution $\mathcal{X}$ as the noise. It is done as follows. We write $\mathbf{A}$ in systematic form via Gaussian elimination. Assume that the first $n$ columns are linearly independent and form the matrix $\mathbf{A_0}$. Define $\mathbf{D} = {\mathbf{A_0}}^{-1}$ and write $\hat{\mathbf{s}} = \mathbf{sD}^{-1} - (z_1, z_2, \ldots, z_n)$. Hence, we can derive an equivalent problem described by $\hat{\mathbf{A}} = (\mathbf{I}, \hat{\mathbf{a}}_{n+1}^{\mathrm{T}}, \hat{\mathbf{a}}_{n+2}^{\mathrm{T}}, \cdots, \hat{\mathbf{a}}_m^{\mathrm{T}})$, where $\hat{\mathbf{A}} = \mathbf{DA}$. We compute

$$\hat{\mathbf{z}} = \mathbf{z} - (z_1, z_2, \ldots, z_n)\hat{\mathbf{A}} = (\mathbf{0}, \hat{z}_{n+1}, \hat{z}_{n+2}, \ldots, \hat{z}_m).$$

Using this transformation, each entry in the secret vector $\mathbf{s}$ is now distributed according to $\mathcal{X}$. The fact that entries in $\mathbf{s}$ are small is a very useful property in several of the

known reduction algorithms for solving LWE.

The noise distribution $\mathcal{X}$ is usually chosen as the discrete Gaussian distribution or the rounded Gaussian Distribution from Section 2.3.

## 3.3 Sample Amplification

In some versions of the LWE problem, such as the Darmstadt Challenges [7], the number of available samples is limited. To get more samples, sample amplification can be used. For example, assume that we have $M$ samples $(\mathbf{a}_1, b_1)$, $(\mathbf{a}_2, b_2)$, ..., $(\mathbf{a}_M, b_M)$. Then we can form new samples, using an index set $I$ of size $k$, as

$$\left(\sum_{j\in I} \pm\mathbf{a}_j, \sum_{j\in I} \pm b_j\right). \tag{3.1}$$

Given an initial number of samples $M$ we can produce up to $2^{k-1}\binom{M}{k}$ samples. This comes at a cost of increasing the noise level (standard deviation) to $\sqrt{k}\cdot\sigma$. This also increases the sample dependency.

## 3.4 Iterating and Guessing

BKW-style algorithms work by combining samples in many steps in such a way that we reach a system of equations over $\mathbb{Z}_q$ of the form $\mathbf{z} = \mathbf{sA} + \mathbf{E}$, where $\mathbf{E} = (E_1, E_2, \ldots, E_m)$ and the entries $E_i, i = 1, 2, \ldots, m$ are sums of not too many original noise vectors, say $E_i = \sum_{j=1}^{2^t} e_{i_j}$, and where $t$ is the number of iterations. The process also reduces the norm of column vectors in $\mathbf{A}$ to be small. Let $n_i, i = 1, 2, \ldots, t$ denote the number of reduced positions in step $i$ and let $N_i = \sum_{j=1}^{i} n_j$. If $n = N_t$, then every reduced equation is of form

$$z_i = \langle \mathbf{a_i}, \mathbf{s}\rangle + E_i, \tag{3.2}$$

for $i = 1, 2, \ldots, m$. The right hand side can be approximated as a sample drawn from a discrete Gaussian and if the standard deviation is not too large, then the sequence of samples $z_1, z_2, \ldots$ can be distinguished from a uniform distribution. We will then need to determine the number of required samples to distinguish between the uniform distribution on $\mathbb{Z}_q$ and $\mathcal{X}_\sigma$. Relying on standard theory from statistics, using either previous work [31] or Bleichenbacher's definition of bias [32], we can find that the required number of samples is roughly

$$C \cdot e^{2\pi\left(\frac{\sigma\sqrt{2\pi}}{q}\right)^2}, \tag{3.3}$$

where $C$ is a small positive constant, whose value was studied in [33]. Initially, an optimal but exhaustive distinguisher was used [34]. While minimizing the sample complexity, it was slow and limited the number of positions that could be guessed. This basic approach was improved in [24], using the FFT. This was in turn a generalization of the corresponding distinguisher for LPN, which used the FWHT [14]. It was shown in [33] that the FFT distinguisher matches the sample complexity of the optimal distinguisher.

## 3.5 Plain BKW

The basic BKW algorithm was originally developed for solving LPN in [13]. It was first applied to LWE in [17]. The reduction part of this approach means that we reduce a fixed number $b$ of positions in the column vectors of $\mathbf{A}$ to zero in each step. In each iteration, the dimension of $\mathbf{A}$ is decreased by $b$ and after $t$ iterations the dimension has decreased by $bt$.

## 3.6 Coded-BKW and LMS

LMS was introduced in [18] and improved in [20]. Coded-BKW was introduced in [19]. Both methods reduce positions in the columns of $\mathbf{A}$ to a small magnitude, but not to zero, allowing reduction of more positions per step. In LMS this is achieved by mapping samples to the same category if the $n_i$ considered positions give the same result when integer divided by a suitable parameter $p$. In coded-BKW this is instead achieved by mapping samples to the same category if they are close to the same codeword in an $[n_i, k_i]$ linear code, for a suitable value $k_i$. Samples mapped to the same category give rise to new samples by subtracting them. The main idea [20, 19] is that positions in later iterations do not need to be reduced as much as the first ones, giving different $n_i$ values in different steps.

## 3.7 LF1, LF2, Unnatural Selection

Each step of the reduction part of the BKW algorithm consists of two parts. First samples are mapped to categories depending on their position values on the currently relevant $n_i$ positions. Next, pairs of samples within the categories are added/subtracted to reduce the current $n_i$ positions to form a new generation of samples. This can be

done in a couple of different ways.

Originally this was done using what is called LF1. Here we pick a representative from each category and form new samples by adding/subtracting samples to/from this sample. This approach makes the final samples independent, but also gradually decreases the sample size. In [14] the approach called LF2 was introduced. Here we add/subtract every possible pair within each category to form new samples. This approach requires only 3 samples within each category to form a new generation of the same size. The final samples are no longer independent, but experiments have shown that this effect is negligible.

In [18] unnatural selection was introduced.The idea is to produce more samples than needed from each category, but only keep the best samples, typically the ones with minimum norm on the current $N_i$ positions in the columns of $\mathbf{A}$.

### 3.8 Coded-BKW with Sieving

When using coded-BKW or LMS, the previously reduced $N_{i-1}$ positions of the columns of $\mathbf{A}$ increase in magnitude with an average factor $\sqrt{2}$ in each reduction step. This problem was addressed in [22] by using unnatural selection to only produce samples that kept the magnitude of the previous $N_{i-1}$ positions small. Instead of testing all possible pairs of samples within the categories, this procedure was sped-up using lattice sieving techniques of [21]. This approach was slightly improved in [11, 23].

## 4 BKW-style Reduction Using Smooth-LMS

In this section we introduce a new reduction algorithm solving the problem of having the same complexity and memory usage in each iteration of a BKW-style reduction. The novel idea is to use simple LMS to reduce a certain number of positions and then partially reduce one extra position. This allows for balancing the complexity among the steps and hence to reduce more positions in total.

### 4.1 A New BKW-style Step

Assume having a large set of samples written as before in the form $\mathbf{z} = \mathbf{sA} + \mathbf{e} \bmod q$. Assume also that the entries of the secret vector $\mathbf{s}$ are drawn from some restricted

distribution with small standard deviation (compared to the alphabet size $q$). If this is not the case, the transformation from Section 3.2 should be applied. Moreover, in case the later distinguishing process involves some positions to be guessed or transformed, we assume that this has been already considered and all positions in our coming description should be reduced.

The goal of this BKW-type procedure is to make the norms of the column vectors of $\mathbf{A}$ small by adding and subtracting equations together in a number of steps. Having expressions of the form $z_i = \mathbf{s}\mathbf{a}_i + E_i \bmod q$, if we can reach a case where $||\mathbf{a}_i||$ is not too large, then $\mathbf{s}\mathbf{a}_i + E_i$ can be considered as a random variable drawn from a discrete Gaussian distribution $\mathcal{X}_\sigma$. Furthermore, $\mathcal{X}_\sigma \bmod q$ can be distinguished from a uniform distribution over $\mathbb{Z}_q$ if $\sigma$ is not too large.

Now let us describe the new reduction procedure. Fix the number of reduction steps to be $t$. We will also fix a maximum list size to be $2^v$, meaning that $\mathbf{A}$ can have at most $2^v$ columns. In each iteration $i$, we are going to reduce some positions to be upper limited in magnitude by $C_i$, for $i = 1, ..., t$. Namely, these positions that are fully treated in iteration $i$ will only have values in the set $\{-C_i + 1, \ldots, 0, 1, \ldots, C_i - 1\}$ of size $2C_i - 1$. We do this by dividing up the $q$ possible values into intervals of length $C_i$. We also adopt the notation $\beta_i = q/C_i$, which describes the number of intervals we divide up the positions into. We assume that $\beta_i > 2$.

**First step.** In the first iteration, assume that we have stored $\mathbf{A}$. We first compute the required compression starting in iteration 1 by computing $C_1$ (we will explain how later). We then evaluate how many positions $n_1$ that can be fully reduced by computing $n_1 = \lfloor v/\log \beta_1 \rfloor$. The position $n_1 + 1$ can be partially reduced to be in an interval of size $C_1'$ fulfilling $\beta_1' \cdot \beta_1^{n_1} \cdot 3/2 \leq 2^v$, where $\beta_1' = q/C_1'$. Now we do an LMS step that "transfers between iterations" in the following way.

We run through all the columns of $\mathbf{A}$. For column $i$, we simply denote it as $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and we compute:

$$k_j = \begin{cases} x_j \text{ div } C_1, & x_1 \geq 0 \\ -x_j \text{ div } C_1, & x_1 < 0 \end{cases}, \quad \text{for} \quad j = 1, \ldots, n_1,$$

$$k_{n_1+1} = \begin{cases} x_{n_1+1} \text{ div } C_1', & x_1 \geq 0 \\ -x_{n_1+1} \text{ div } C_1', & x_1 < 0 \end{cases}.$$

The vector $K_i = (k_1, k_2, \ldots, k_{n_1+1})$ is now an index to a sorted list $\mathcal{L}$, storing these

vectors[1]. Except for the inverting of values if $x_1 < 0$, samples should have the same index if and only if all position values are the same when integer divided by $C_1$ ($C'_1$ for the last position). So we assign $\mathcal{L}(K_i) = \mathcal{L}(K_i) \cup \{i\}$. After we have inserted all columns into the list $\mathcal{L}$, we go to the combining part.

We build a new matrix $\mathbf{A}$ in the following way. Run through all indices $K$ and if $|\mathcal{L}(K)| \geq 2$ combine every pair of vectors in $\mathcal{L}(K)$ by subtracting/adding[2] them to form a new column in the new matrix $\mathbf{A}$. Stop when the number of new columns has reached $2^v$. For each column in $\mathbf{A}$ we have that:

- the absolute value of each position $j \in \{1, \ldots, n_1\}$ is $< C_1$,
- the absolute value of position $n_1 + 1$ is $< C'_1$.

**Next steps.** We now describe all the next iterations, numbered as $l = 2, 3, \ldots, t$. Iteration $l$ will involve positions from $N_{l-1} + 1 = \sum_{i=1}^{l-1} n_i + 1$ to $N_l + 1$. The very first position has possibly already been partially reduced and its absolute value is $< C'_{l-1}$, so the interval for possible values is of size $2C'_{l-1} - 1$. Assume that the desired interval size in iteration $l$ is $C_l$. In order to achieve the corresponding reduction factor $\beta_l$, we split this interval in $\beta''_l = (2C'_{l-1} - 1)/C_l$ subintervals. We then compute how many positions $n_l$ that can be fully reduced by computing $n_l = \lfloor (v - \log \beta''_l)/\log \beta_l \rfloor$. The position $N_l + 1$ can finally be partially reduced to be in an interval of size $C'_l$ fulfilling $\beta'_l \cdot \beta_l^{n_l - 1} \beta''_l \cdot 3/2 \leq 2^v$, where $\beta'_l = q/C'_l$.

Similar to iteration 1, we run through all the columns of $\mathbf{A}$. For each column $i$ in the matrix $\mathbf{A}$ denoted as $\mathbf{x}$ we do the following. For each vector position in $\{N_{l-1} + 1, \ldots, N_l + 1\}$, we compute (here div means integer division)

$$
\begin{aligned}
k_j &= \begin{cases} x_{N_{l-1}+j} \text{ div } C_l, & x_{N_{l-1}+1} \geq 0 \\ -x_{N_{l-1}+j} \text{ div } C_l, & x_{N_{l-1}+1} < 0 \end{cases}, \quad \text{for} \quad j = 1, \ldots, n_l, \\
k_{n_l} &= \begin{cases} x_{N_l+1} \text{ div } C'_l, & x_{N_{l-1}+1} \geq 0 \\ -x_{N_l+1} \text{ div } C'_l, & x_{N_{l-1}+1} < 0 \end{cases}.
\end{aligned} \tag{4.1}
$$

The vector $K = (k_1, k_2, \ldots, k_{n_{l+1}})$ is again an index to a sorted list $\mathcal{L}$, keeping track of columns[3]. So again we assign $\mathcal{L}(K) = \mathcal{L}(K) \cup \{i\}$. After we have inserted all column

[1] The point of inverting all position values if $x_1 < 0$ is to make sure that samples that get reduced when added should be given the same index. For example $(x_1, x_2, \ldots, x_{n_1+1})$ and $(-x_1, -x_2, \ldots, -x_{n_1+1})$ are mapped to the same category.

[2] Depending on what reduces the sample the most.

[3] Also here the point of inverting all position values if $x_{N_{l-1}+1} < 0$ is to make sure that samples that get reduced when added should be given the same index. For example $(x_{N_{l-1}+1}, x_{N_{l-1}+2}, \ldots, x_{N_l+1})$

indices into the list $\mathcal{L}$, we go to the combining part.

As in the first step, we build a new $\mathbf{A}$ as follows. Run through all indices $K$ and if $|\mathcal{L}(K)| \geq 2$ combine every pair of vectors by adding/subtracting them to form a column in the new matrix $\mathbf{A}$. Stop when the number of new columns has reached $2^v$.

For the last iteration, since $N_t$ is the last row of $\mathbf{A}$, one applies the same step as above but without reducing the extra position. After $t$ iterations, one gets equations on the form (3.2), where the $\mathbf{a}_i$ vectors in $\mathbf{A}$ have reduced norm.

## 4.2 Smooth-Plain BKW

The procedure described above also applies to plain BKW steps. For example, if in the first iteration one sets $C_1 = 1$ and $C_1' > 1$, then each column vector $\mathbf{x}$ of $\mathbf{A}$ will be reduced such that $x_1 = \ldots = x_{n_1} = 0$ and $x_{n_1+1} \in \{-C_1' + 1, \ldots, C_1' - 1\}$. Thus, one can either continue with another smooth-Plain BKW step by setting also $C_2 = 1$ in the second iteration, or switch to smooth-LMS. In both cases, we have the advantage of having $x_{n_1}$ already partially reduced. Using these smooth-Plain steps we can reduce a couple of extra positions in the plain pre-processing steps of the BKW algorithm.

## 4.3 How to Choose the Interval Sizes $C_i$

To achieve as small norm of the vectors as possible, we would like the variance of all positions to be equally large, after completing all iterations. Assume that a position $x$ takes values uniformly in the set $\{-(C-1)/2, \ldots, 0, 1, \ldots, (C-1)/2\}$, for $C > 0$. Thus, we have that $\mathrm{Var}[x] = (C-1)(C+1)/12$. Assuming $C$ is somewhat large, we approximately get $\mathrm{Var}[x] = C^2/12$. When subtracting/adding two such values, the variance increases to $2\mathrm{Var}[x]$ in each iteration. Therefore, a reduced position will have an expected growth of $\sqrt{2}$. For this reason, we choose a relation for the interval sizes of the form

$$C_i = 2^{-(t-i)/2} C_t, \quad i = 1, \ldots, t-1.$$

This makes the variance of each position roughly the same, after completing all iterations. In particular, our vectors $||\mathbf{a}_i||$ in $\mathbf{A}$ are expected to have norm at most $\sqrt{n}C_t/\sqrt{12}$, and $C_t$ is determined according to the final noise allowed in the guessing phase. Ignoring the pre-processing step with smooth-Plain BKW steps, the maximum dimension $n$ that can be reduced is then $n = N_t = \sum_{i=1}^{t} n_i$.

and $(-x_{N_{l-1}+1}, -x_{N_{l-1}+2}, \ldots, -x_{N_l+1})$ are mapped to the same category.

**Example 1.** *Let* $q = 1601$ *and* $\alpha = 0.005$*, so* $\sigma = \alpha q \approx 8$*. Let us compute how many positions can be reduced using* $2^v = 2^{28}$ *list entries. The idea is that the variance of the right hand side in (3.2) should be minimized by making the variance of the two terms roughly equal. The error part* $E_i$ *is the sum of* $2^t$ *initial errors, so its variance is* $\mathrm{Var}[E_i] = 2^t\sigma^2$*. In order to be able to distinguish the samples according to (3.3), we set* $\mathrm{Var}[E_i] < q^2/2$*. This will give us the number of iterations possible as* $2^t\sigma^2 \approx q^2/2$ *or* $2^t \approx 1601^2/(2 \cdot 8^2)$ *leading to* $t = 14$*. Now we bound the variance of the scalar product part of (3.2) also to be* $< q^2/2$*, so* $n\sigma^2 C_t^2/12 \approx q^2/2$ *leading to* $C_t^2 \approx 12q^2/(2n\sigma^2)$ *and* $C_t^2 \approx 12 \cdot 1601^2/(2n \cdot 8^2)$ *or* $C_t \approx 80$ *if* $n < 38$*. Then one chooses* $C_{t-1} = \lfloor C_t/\sqrt{2} \rceil = 57$ *and so on.*

## 4.4 Unnatural Selection

We can improve performance by using the unnatural selection discussed in Section 3.7. Let us make some basic observations. Combining $n_l$ positions using interval size $C$ gives as previously described a value in the set $\{-(C-1)/2, \ldots, 0, 1, \ldots (C-1)/2\}$, and results in $\mathrm{Var}[x] = (C-1)(C+1)/12$. Combining two vectors from the same category, a position value $y = x_1 + x_2$, where $x_1, x_2$ are as above, results in a value in the interval $\{-(C-1), \ldots, 0, 1, \ldots (C-1)\}$ with variance $\mathrm{Var}[y] = (C-1)(C+1)/6$. Now observe that for the resulting reduced positions, smaller values are much more probable than larger ones.

## 4.5 On Optimizing $C_l$ Values

The choice of the parameter $C_l$ within a Smooth-LMS step can be optimized in order to achieve a lower number of category in the next step. For example, consider $q = 1601$ and $C_l = 250$. The number of categories for a single position would be $\left\lfloor \frac{q}{2C_l+1} \right\rfloor = 3$. Clearly, the same result can be obtained if one chooses $C_l = 200$. The difference is that with this second choice of $C_l$, for the same cost (linear on the number of categories), one gets more reduced samples at the end of the step. Therefore, a lower number of categories is required for the next step.

## 4.6 An Illustration of Smooth Reduction Steps

Figure 3.1 illustrates how the smooth versions of LMS steps and plain BKW reduction steps outperforms their standard counterparts by partially reducing an extra position in

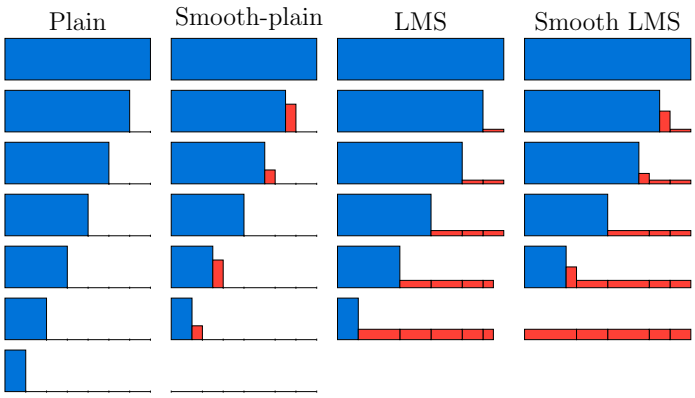


Figure 3.1: An illustration of how the magnitudes of the $\mathbf{a}$ vectors change over the reduction steps for different versions of the BKW algorithm. The heights correspond to magnitudes and the width corresponds to the number of positions. The figure is taken from [35].

each step. The figure is from [35]. In [35] Figures 7.1 and 7.3 also illustrate aspects of BKW reduction algorithms.

# 5 A Binary Partial Guessing Approach

In this section we propose a new way of reducing the guessing step to a binary version. This way, we are able to efficiently use the FWHT to guess many entries in a small number of operations. In Section 6 we do the theoretical analysis and show that this indeed leads to a more efficient procedure than all previous ones.

## 5.1 From LWE to LPN

First, we need to introduce a slight modification to the original system of equations *before* the reduction part. Assume that we have turned the distribution of $\mathbf{s}$ to be the noise distribution, through the standard transformation described in Section 3.2. The result after this is written as before

$$\mathbf{z} = \mathbf{sA} + \mathbf{e}. \tag{5.1}$$

Now we perform a multiplication by 2 to each equation, resulting in

$$\mathbf{z}' = \mathbf{s}\mathbf{A}' + 2\mathbf{e},$$

since when multiplied with a known value, we can compute the result modulo $q$.

Next, we apply the reduction steps and make the values in $\mathbf{A}'$ as small as possible by performing BKW-like steps. In our case we apply the smooth-LMS step from the previous section, but any other reduction method like coded-BKW with sieving would be possible. If $\mathbf{A}' = \begin{bmatrix}\mathbf{a}_1^{\mathrm{T}} & \mathbf{a}_2^{\mathrm{T}} & \cdots & \mathbf{a}_m^{\mathrm{T}}\end{bmatrix}$ the output of this step is a matrix where the Euclidean norm of each $\mathbf{a}_i$ is small. The result is written as

$$\mathbf{z}'' = \mathbf{s}\mathbf{A}'' + 2\mathbf{E}, \tag{5.2}$$

where $\mathbf{E} = (E_1, E_2, \ldots, E_m)$ and $E_i = \sum_{j=1}^{2^t} e_{i_j}$ as before.

Finally, we transform the entire system to the binary case by considering

$$\mathbf{z}_0'' = \mathbf{s}_0\mathbf{A}_\mathbf{0}'' + \mathbf{e} \bmod 2, \tag{5.3}$$

where $\mathbf{z}_0''$ is the vector of least significant bits in $\mathbf{z}''$, $\mathbf{s}_0$ the vector of least significant bits in $\mathbf{s}$, $\mathbf{A}_\mathbf{0}'' = (\mathbf{A}'' \bmod 2)$ and $\mathbf{e}$ denotes the binary error introduced.

We can now examine the error $e_j$ in position $j$ of $\mathbf{e}$. In (5.2) we have equations of the form $z_j = \sum_i s_i a_{ij} + 2E_j$ in $\mathbb{Z}_q$, which can be written on integer form as

$$z_j = \sum_i s_i a_{ij} + 2E_j + k_j \cdot q. \tag{5.4}$$

Now if $|\sum_i s_i a_{ij} + 2E_j| < q/2$ then $k_j = 0$. In this case (5.4) can be reduced mod 2 without error and $e_j = 0$. In general, the error is computed as $e_j = k_j \bmod 2$. So one can compute a distribution for $e_j = k_j \bmod 2$ by computing $P(k_j = x)$. It is possible to compute such distribution either making a general approximation or precisely for each specific position $j$ using the known values $\mathbf{a}_j$ and $z_j$. Note that the distribution of $e_j$ depends on $z_j$. Also note that if $\mathbf{a}_j$ is reduced to a small norm and the number of steps $t$ is not too large, then it is quite likely that $|\sum_i s_i a_{ij} + 2E_j| < q/2$ leading to $P(e_j = 0)$ being large.

For the binary system, we finally need to find the secret value $\mathbf{s}_0$. Either

1. there are no errors (or almost no errors), corresponding to $P(e_j = 0) \approx 1$. Then one can solve for $\mathbf{s}_0$ directly using Gaussian elimination (or possibly some information set decoding algorithm in the case of a few possible errors).
2. or the noise is larger. The binary system of equations corresponds to the situation of a fast correlation attack [36], or secret recovery in an LPN problem [13]. Thus, one may apply an FWHT to recover the binary secret values.

## 5.2 Guessing $\mathbf{s}_0$ Using the FWHT

The approach of using the FWHT to find the most likely $\mathbf{s}_0$ in the binary system in (5.3) comes directly from previous literature on Fast Correlation Attacks [37].

Let $k$ denote an $n$-bit vector $(k_0, k_1, \ldots, k_{n-1})$ (also considered as an integer) and consider a sequence $X_k, k = 0, 1, \ldots, N-1$, $N = 2^n$. It can for example be a sequence of occurrence values in the time domain, e.g. $X_k =$ the number of occurrences of $X = k$. The Walsh-Hadamard Transform is defined as

$$\hat{X}_w = \sum_{k=0}^{N-1} X_k \cdot (-1)^{w \cdot k}, \tag{5.5}$$

where $w \cdot k$ denotes the bitwise dot product of the binary representation of the $n$-bit indices $w$ and $k$. There exists an efficient method (FWHT) to compute the WHT in time $O(N \log N)$. Given the matrix $\mathbf{A}''_{\mathbf{0}}$, we define $X_k = \sum_{j \in J} (-1)^{z''_j}$, where $J$ is the set of all columns of the matrix $\mathbf{A}''_{\mathbf{0}}$ that equal $k$. Then, one computes $\max_w |\hat{X}_w|$, and we have that $\mathbf{s}_0$ corresponds to $\bar{w}$ such that $|\hat{X}_{\bar{w}}| = \max_w |\hat{X}_w|$. In addition, $\hat{X}_w$ is simply the (biased) sum of the noise terms.

### Soft Received Information

The bias of $\hat{X}_w$ actually depends on the value of $z''_j$. So a slightly better approach is to use "soft received information" by defining $X_k = \sum_{j \in J} (-1)^{z''_j} \cdot \varepsilon_{z''_j}$, where $\varepsilon_{z''_j}$ is the bias corresponding to $z''_j$. For each $x \in \{-(q-1)/2, ..., (q-1)/2\}$, the bias $\varepsilon_x$ can be efficiently pre-computed so that its evaluation does not affect the overall complexity of the guessing procedure.

#### Hybrid Guessing

One can use hybrid approach to balance the overall complexity among reduction and guessing phases. Indeed, it is possible to leave some rows of the matrix $\mathbf{A}$ unreduced and apply an exhaustive search over the corresponding positions in combination with the previously described guessing step. Since the overall complexity of the algorithm is additive in reduction and guessing phases, one can use hybrid approach to balance the overall complexity among the two. Moreover, we remark that this exhaustive search can easily benefit from parallelization.

#### Even Selection

When transforming the system to a binary (5.3), we can zero out some positions to get an easier problem. This can be achieved by ensuring that, when reducing $\mathbf{A}'$, some specific rows of $\mathbf{A}''$ are small[4] and additionally have even coefficients, making sure to have enough samples left for the guessing phase. In this way we cancel out the corresponding entries of $\mathbf{s}$ when reducing modulo 2 and get a smaller binary system of equations.

## 5.3 Retrieving the Original Secret

Once $\mathbf{s}_0$ is correctly guessed, it is possible to obtain a new LWE problem instance with the secret half as big as follows. Write $\mathbf{s} = 2\mathbf{s}' + \mathbf{s}_0$. Define $\hat{\mathbf{A}} = 2\mathbf{A}$ and $\hat{\mathbf{z}} = \mathbf{z} - \mathbf{s}_0\mathbf{A}$. Then we have that

$$\hat{\mathbf{z}} = \mathbf{s}'\hat{\mathbf{A}} + \mathbf{e}. \tag{5.6}$$

The entries of $\mathbf{s}'$ have a bit-size half as large as the entries of $\mathbf{s}$, therefore (5.6) is an easier problem than (5.1). One can apply the procedure described above to (5.6) and guess the new binary secret $\mathbf{s}_1$, i.e. the least significant bits of $\mathbf{s}'$. The cost of doing this will be significantly smaller as shorter secret translates to computationally easier reduction steps. Thus, computationally speaking, the LWE problem can be considered solved once we manage to guess the least significant bits of $\mathbf{s}$. Given the list of binary vectors $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_d$, it is easy to retrieve the original secret $\mathbf{s}$.

Generally, if $s_{d_i} = 0$, then $s_i \geq 0$ and $(s_{0_i}, s_{1_i}, ..., s_{d_i})$ is nothing else than its binary representation. Conversely, if $s_{d_i} = 1$, then $s_i < 0$. To compute its magnitude in this case, one must look again at $(s_{0_i}, s_{1_i}, ..., s_{d_i})$ and consider that all negative entries of $\mathbf{s}$

[4]For a specific entry $a_j$ in a vector and the corresponding value $s_j$ in the secret, we should have $a_j \cdot s_j < q$.

**Algorithm 2:** BKW-FWHT with smooth reduction (main framework)

**Input:** Matrix $\mathbf{A}$ with $n$ rows and $m$ columns, received vector $\mathbf{z}$ of length $m$ and algorithm parameters $t_1, t_2, t_3, n_{limit}, \sigma_{set}$
Step 0: Use Gaussian elimination to change the distribution of the secret vector;
Step 1: Use $t_1$ smooth-plain BKW steps to remove the bottom $n_{pbkw}$ entries;
Step 2: Use $t_2$ smooth-LMS steps to reduce $n_{cod1}$ more entries;
Step 3: Perform the multiplying-2 operations;
Step 4: Use $t_3$ smooth-LMS steps to reduce the remaining $n_t \leq n_{limit}$ entries;
Step 5: Transform all the samples to the binary field and recover the partial secret key by the FWHT. We can exhaustively guess some positions.

cannot be reduced any further than $-1$. Namely, if for example $s_k$ is negative and at step $j < d$ is reduced to be $-1$, then $s_{j_k} = s_{(j+1)_k} = \cdots = s_{d_k} = 1$.

The following example shows how to retrieve the original secret $\mathbf{s}$ once the list of least significant bits vectors $\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_d$ has been guessed.

**Example 2.** *Assume that the secret has length $n = 10$ and that its entries' distribution has standard deviation $\sigma = 3$. Then, performing the above procedure $\lceil \log_2 4\sigma \rceil = 4$ times, with high probability we reduced the secret as much as possible. In the following example, one can note that $\mathbf{s}_3$ determines the sign of $\mathbf{s}$. Therefore, the magnitude of $s_i$ is retrieved by looking at $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2$.*

$$\begin{array}{lcccccccccc} \mathbf{s}_0: & ( & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & ) \\ \mathbf{s}_1: & ( & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & ) \\ \mathbf{s}_2: & ( & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & ) \\ \mathbf{s}_3: & ( & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & ) \\ \hline \mathbf{s}: & ( & -3 & 0 & 2 & 0 & 1 & -1 & 4 & 3 & -2 & -4 & ) \end{array}$$

*Note that if one performs one more iteration, the corresponding binary secret $\mathbf{s}_4$ will be the same as $\mathbf{s}_3$ because we already reached the maximum reduction of $\mathbf{s}$ possible.*

# 6 Analysis of the Algorithm and its Complexity

In this section, we describe in detail the newly-proposed algorithm called BKW-FWHT with smooth reduction (BKW-FWHT-SR).

## 6.1 The Algorithm

The main steps of the new BKW-FWHT-SR algorithm are described in Algorithm 2. We start by changing the distribution of the secret vector with the secret-noise transformation [29], if necessary.

The general framework is similar to the coded-BKW with sieving procedure proposed in [22]. In our implementation, we instantiated coded-BKW with sieving steps with smooth-LMS steps discussed before for the ease of implementation.

The different part of the new algorithm is that after certain reduction steps, we perform a multiplication by 2 to each reduced sample as described in Section 5. We then continue reducing the remain positions and perform the mod 2 operations to transform the entire system to the binary case. Now we obtain a list of LPN samples and solve the corresponding LPN instance via known techniques such as FWHT or partial guessing.

One high level description is that we aim to input an LWE instance to the LWE-to-LPN transform developed in Section 5, and solve the instance by using a solver for LPN. To optimize the performance, we first perform some reduction steps to have a new LWE instance with reduced dimension but larger noise. We then feed the obtained instance to the LWE-to-LPN transform.

## 6.2 The Complexity of Each Step

From now on we assume that the secret is already distributed as the noise distribution or that the secret-noise transform is performed. We use the LF2 heuristics and assume the sample size is unchanged before and after each reduction step. We now start with smooth-plain BKW steps and let $l_{red}$ be the number of positions already reduced.

#### Smooth-Plain BKW steps.

Given $m$ initial samples, we could on average have $\lfloor \frac{2m}{3} \rfloor$ categories[5] for one plain BKW step in the LF2 setting. Instead we could assume for $2^{b_0}$ categories, and thus the number of samples $m$ is $1.5 \cdot 2^{b_0}$. Let $C_{pBKW}$ be the cost of all smooth-plain BKW steps, whose initial value is set to be 0. If a step starts with a position never being reduced before, we can reduce $l_p$ positions, where $l_p = \left\lfloor \frac{b}{\log_2(q)} \right\rfloor$. Otherwise, when the first position is

[5]The number of categories is halved compared with the LF2 setting for LPN. The difference is that we could add and subtract samples for LWE.

partially reduced in the previous step and we need $\beta'$ categories to further reduce this position, we can in total fully reduce $l_p$ positions, where $l_p = 1 + \left\lfloor \frac{b-\log_2(\beta')}{\log_2(q)} \right\rfloor$.

For this smooth-plain BKW step, we compute

$$C_{pbkw} \mathrel{+}= ((n + 1 - l_{red}) \cdot m + C_{d,pbkw}),$$

where $C_{d,pbkw} = m$ is the cost of modulus switching for the last partially reduced position in this step. We then update the number of the reduced positions, $l_{red} \mathrel{+}= l_p$.

After iterating for $t_1$ times, we could compute $C_{pbkw}$ and $l_{red}$. We will continue updating $l_{red}$ and denote $n_{pbkw}$ the length reduced by the smooth-plain BKW steps.

**Smooth-LMS steps before the multiplication of 2.**

We assume that the final noise contribution from each position reduced by LMS is similar, bounded by a preset value $\sigma_{set}$. Since the noise variable generated in the $i$-th $(0 \le i \le t_2 - 1)$ Smooth-LMS step will be added by $2^{t_2+t_3-i}$ times and also be multiplied by 2, we compute $\sigma^2_{set} = \frac{2^{t_2+t_3-i} \times 4C^2_{i,LMS1}}{12}$, where $C_{i,LMS1}$ is the length of the interval after the LMS reduction in this step. We use $\beta_{i,LMS1}$ categories for one position, where $\beta_{i,LMS1} = \lceil \frac{q}{C_{i,LMS1}} \rceil$. Similar to smooth-plain BKW steps, if this step starts with an new position, we can reduce $l_p$ positions, where $l_p = \lfloor \frac{b}{\log_2(\beta_{i,LMS1})} \rfloor$. Otherwise, when the first position is partially reduced in the previous step and we need $\beta'_{p,i,LMS1}$ categories to further reduce this position, we can in total fully reduce $l_p$ positions, where $l_p = 1 + \lfloor \frac{b-\log_2(\beta'_{p,i,LMS1})}{\log_2(\beta_{i,LMS1})} \rfloor$. Let $C_{LMS1}$ be the cost of Smooth-LMS steps before the multiplication of 2, which is innitialized to 0. For this step, we compute

$$C_{LMS1} \mathrel{+}= (n + 1 - l_{red}) \cdot m,$$

and then update the number of the reduced positions, $l_{red} \mathrel{+}= l_p$.

After iterating $t_2$ times, we compute $C_{LMS1}$ and $l_{red}$. We expect $l_{red} = n - n_t$ $(n_t \le n_{limit})$ positions have been fully reduced and will continue updating $l_{red}$.

**Smooth-LMS steps after the multiplication of 2.**

The formulas are similar to those for Smooth-LMS steps before the multiplication of 2. The difference is that the noise term is no longer multiplied by 2, so we have $\sigma^2_{set} = \frac{2^{t_3-i} C^2_{i,LMS2}}{12}$, for $0 \le i \le t_3 - 1$. Also, we need to track the $a$ vector of length $n_t$ for the

later distinguisher. The cost is

$$C_{LMS2} = t_3 \cdot (n_t + 1) \cdot m.$$

We also need to count the cost for multiplying samples by 2 and the mod2 operations, and the LMS decoding cost, which are

$$C_{mulMod} = 2 \cdot (n_t + 1) \cdot m,$$
$$C_{dec} = (n - n_{pbkw} + t_2 + t_3) \cdot m.$$

**FWHT distinguisher and partial guessing.**

After the LWE-to-LPN transformation, we have an LPN problem with dimension $n_t$ and $m$ instance. We perform partial guessing on $n_{guess}$ positions, and use FWHT to recover the remaining $n_{FWHT} = n_t - n_{guess}$ positions. The cost is,

$$C_{distin} = 2^{n_{guess}} \cdot ((n_{guess} + 1) \cdot m + n_{FWHT} \cdot 2^{n_{FWHT}}).$$

## 6.3 The Data Complexity

We now discuss the data complexity of the new FWHT distinguisher. In the integer form, we have the following equation,

$$z_j = \sum_{i=0}^{n_t - 1} s_i a_{ij} + 2E_j + k_j \cdot q.$$

If $|\sum s_i a_{ij} + 2E_j| < q/2$ then $k_j = 0$. Then the equation can be reduced mod 2 without error. In general, the error is $e_j = k_j \bmod 2$.

We employ a smart FWHT distinguisher with soft received information, as described in Section 5. From [38], we know the sample complexity can be approximated as $m \approx \frac{4 \ln(2^{n_t})}{\mathbb{E}_{z=t}[D(e_{z=t}||U_b)]}$.

For different value of $z_j$, the distribution of $e_j$ is different. The maximum bias is

| $\sigma_f$ | q | $D(\mathcal{X}_{\sigma_f,2q}\|\|U_{2q})$ | $\mathbb{E}_{z=t}[D(e_{z=t}\|\|U_b)]$ |
|---|---|---|---|
| $0.5q$ | 1601 | $-2.974149$ | $-2.974995$ |
| $0.6q$ | 1601 | $-4.577082$ | $-4.577116$ |
| $0.7q$ | 1601 | $-6.442575$ | $-6.442576$ |
| $0.8q$ | 1601 | $-8.582783$ | $-8.582783$ |

Table 6.1: The comparison between $D(\mathcal{X}_{\sigma_f,2q}||U_{2q})$ and $\mathbb{E}_{z=t}[D(e_{z=t}||U_b)]$

achieved when $z_j = 0$. In this sense, we could compute the divergence as

$$\begin{aligned}\mathbb{E}_{z=t}[D(e_{z=t}||U_b)] &= \sum_{t\in\mathbb{Z}_q} Pr\,(z=t)\, D(e_{z=t}||U_b) \\ &= \sum_{t\in\mathbb{Z}_q} Pr\,(z=t)\,(\sum_{i=0}^{1} Pr\,(e_{z=t}=i)\log(2\cdot Pr\,(e_{z=t}=i)))\end{aligned}$$

where $e_z$ is the Bernoulli variable conditioned on the value of $z$ and $U_b$ the uniform distribution over the binary field.

Following the previous research [17], we approximate the noise $\sum s_i a_{ij} + 2E_j$ as discrete Gaussian with standard deviation $\sigma_f$. If $\sigma_f$ is large, the probability $Pr\,(z=t)$ is very close to $1/q$. Then, the expectation $\mathbb{E}_{z=t,t\in\mathbb{Z}_q}[D(e_{z=t}||U_b)]$ can be approximated as

$$\sum_{t\in\mathbb{Z}_q}\sum_{i=0}^{1} Pr\,(z=t)\, Pr\,(e_{z=t}=i)\log(2q\cdot Pr\,(e_{z=t}=i,z=t)),$$

i.e., the divergence between a discrete Gaussian with the same standard deviation and a uniform distribution over $2q$, $D(\mathcal{X}_{\sigma_f,2q}||U_{2q})$. We numerically computed that the approximation is rather accurate when the noise is sufficiently large (see Table 6.1). In conclusion, we use the formula

$$m \approx \frac{4\ln(2^{n_t})}{D(\mathcal{X}_{\sigma_f,2q}||U_{2q})},$$

to estimate the data complexity of the new distinguisher. It remains to control the overall variance $\sigma_f^2$. Since we assume that the noise contribution from each reduced position by LMS is the same and the multiplication of 2 will double the standard deviation, we can derive $\sigma_f^2 = 4 * 2^{t_1+t_2+t_3}\sigma^2 + \sigma^2\sigma_{set}^2(n - n_{pbkw})$.

**Note:** The final noise is a combination of three parts, the noise from the LWE problem, the LMS steps before the multiplication by 2, and the LMS steps after the multiplication by 2. The final partial key recovery problem is equivalent to distinguishing a discrete

Gaussian from uniform with the alphabet size doubled. We see that with the multiplication by 2, the variances of the first and the second noise parts are increased by a factor of 4, but the last noise part does not expand. This intuitively explains the gain of the new binary distinguisher.

## 6.4 In Summary

We have the following theorem to estimate the complexity of the attack.

**Theorem 1.** *The time complexity of the new algorithm is*

$$C = C_{pbkw} + C_{LMS1} + C_{LMS2} + C_{dec} + C_{distin} + C_{mulMod},$$

*under the condition that*

$$m \geq \frac{4\ln(2^{n_t})}{D(\mathcal{X}_{\sigma_f,2q}||U_{2q})},$$

*where* $\sigma_f^2 = 4 * 2^{t_1+t_2+t_3}\sigma^2 + \sigma^2\sigma_{set}^2(n - n_{pbkw})$.

## 6.5 Numerical Estimation

We numerically estimate the complexity of the new algorithm BKW-FWHT-SR (shown in Table 6.2). It improves the known approaches when the noise rate (represented by $\alpha$) becomes larger. We should note that compared with the previous BKW-type algorithms, the implementation is much easier though the complexity gain might be mild.

# 7 Implementations

We produced two different C implementations of the BKW algorithm, as presented in this manuscript. These mainly differ in memory management. The first one referred to as RBBL (Ram-Based BKW for LWE), is light, fast, and relies only on RAM usage. However, this turned out to be a limiting factor for solving hard LWE instances. Therefore, the second implementation, referred to as FBBL (File-Based BKW for LWE), follows a design strategy that can be seen as a transition away from such a limiting factor. The main idea is to use a file-based approach to store the samples, moving the memory constraint from RAM to disk capacity. More details are given in the next sections. Both implementations are available as open-source libraries at https://github.com/FBBL. Some simulation results are presented in Section 8.

| $n$ | $q$ | $\alpha$ | | LWE estimator [9] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | BKW-FWHT-SR | Coded-BKW | usvp ENU | usvp Sieve | dec ENU | dec Sieve | dual ENU | dual Sieve |
| 40 | 1601 | 0.005 | 34.4 | 42.6 | **31.4** | 41.5 | 34.7 | 44.6 | 39.1 | 47.5 |
| | | 0.010 | 39.3 | 43.7 | **34.0** | 44.8 | 36.3 | 44.9 | 51.1 | 57.9 |
| | | 0.015 | **42.4** | 52.6 | 42.5 | 54.2 | 43.1 | 50.6 | 61.5 | 64.4 |
| | | 0.020 | **46.2** | 52.6 | - | - | 51.9 | 58.2 | 73.1 | 75.9 |
| | | 0.025 | **48.3** | 52.7 | - | - | 59.2 | 66.1 | 84.7 | 85.4 |
| | | 0.030 | **50.0** | 52.7 | - | - | 67.1 | 68.9 | 96.3 | 92.5 |
| 45 | 2027 | 0.005 | 37.7 | 55.2 | **31.8** | 41.9 | 35.0 | 44.8 | 41.5 | 51.6 |
| | | 0.010 | 43.5 | 55.2 | **39.5** | 51.2 | 41.2 | 48.2 | 57.0 | 64.6 |
| | | 0.015 | **48.3** | 55.2 | 50.4 | 61.3 | 51.2 | 58.3 | 74.3 | 74.9 |
| | | 0.020 | **51.2** | 55.2 | - | - | 61.1 | 65.0 | 86.8 | 86.1 |
| | | 0.025 | **54.1** | 55.3 | - | - | 71.0 | 71.4 | 100.7 | 95.0 |
| | | 0.030 | **56.3** | 64.1 | - | - | 80.2 | 78.7 | 116.2 | 104.1 |
| 50 | 2503 | 0.005 | 41.8 | 46.4 | **32.4** | 42.6 | 35.5 | 45.1 | 46.7 | 58.0 |
| | | 0.010 | 48.7 | 56.0 | **46.0** | 57.5 | 47.6 | 54.1 | 66.8 | 65.4 |
| | | 0.015 | **52.5** | 56.8 | - | - | 60.8 | 63.6 | 84.9 | 83.5 |
| | | 0.020 | **56.4** | 61.9 | - | - | 72.1 | 72.1 | 101.9 | 96.5 |
| | | 0.025 | **59.3** | 66.1 | - | - | 83.5 | 80.8 | 120.0 | 105.7 |
| | | 0.030 | **63.3** | 66.3 | - | - | 94.2 | 89.1 | 134.0 | 115.6 |
| 70 | 4903 | 0.005 | 58.3 | 62.3 | **52.3** | 54.2 | 55.2 | 63.3 | 76.2 | 75.9 |
| | | 0.010 | **67.1** | 73.7 | - | - | 80.4 | 77.1 | 111.3 | 98.9 |
| | | 0.015 | **73.3** | 75.6 | - | - | 102.5 | 93.2 | 146.0 | 118.0 |
| 120 | 14401 | 0.005 | 100.1 | 110.5 | 133.0 | **93.2** | 135.5 | 111.4 | 181.9 | 133.2 |
| | | 0.010 | **115.1** | 124.0 | - | - | 195.0 | 150.4 | 266.2 | 165.7 |
| | | 0.015 | **127.0** | 136.8 | - | - | 246.4 | 183.2 | 334.0 | 209.8 |

Table 6.2: Estimated time complexity comparison (in $\log_2(\cdot)$) for solving LWE instances in the TU Darmstadt LWE challenge [7]. Here unlimited number of samples are assumed. The last columns show the complexity estimation from the LWE estimator [9]. "ENU" represents the enumeration cost model is employed and "Sieve" represents the sieving cost model is used. Bold-faced numbers are the smallest among the estimations with these different approaches.

## 7.1 RAM-based Implementation

Storing samples directly in RAM allows a simple and fast implementation. The purpose of RBBL is to achieve good results in solving relatively small LWE instances and provide a comparison against the file-based solution FBBL. RBBL supports Smooth-

LMS reduction steps, including Smooth-Plain BKW steps. The FWHT-based guessing method introduced in Sections 4 and 5 is implemented in both its plain and hybrid version.

#### Memory and sample organization

The samples are stored in *heap* within a single array, allowing for a single (and faster) memory allocation. For each category, we allocate enough memory to fit a fixed number of samples. Therefore, if for a certain category the number of new samples exceeds the category capacity, some samples get discarded.

#### Parallelization

RBBL supports parallelization using the *POSIX Threads* utility. Within the steps of the reduction phase, each thread gets assigned to different sets of categories and performs independently sums and subtractions of samples. A system of mutexes prevents memory corruption caused by two or more threads writing to the same category, and therefore to the same memory cells, at the same time. The guessing phase benefits from parallelization when using the hybrid guesser. All the possible combinations of entries of the bruteforce positions are equally distributed among the available threads, which perform an FWHT for each one of them. Finally, one chooses the guess paired with the highest probability, as explained in Section 5.2.

## 7.2 File-based Implementation

A file-based implementation is needed when the amount of memory required to store the samples exceeds the available RAM. FBBL supports most known BKW reduction steps, FFT and FWHT-based guessing methods, and hybrid guessing approaches. Different reduction steps can be combined arbitrarily, and the implementation also allows full recovery of the initial secret, including the initial transform and guessing all parts of the secret vector (if all intermediate reduction results have been stored and are compatible). A key success factor in the software design was to avoid unnecessary reliance on RAM, so we have employed file-based storage where necessary and practically possible. We describe below how we dealt with some interesting performance issues.

### File-based Sample Storage

Samples get stored in a file, sorted according to their category. The necessary space for a fixed number of samples is reserved into the file for each category. Then a *storage writer* writes down the samples from RAM to file in the space reserved for their categories (possibly discarding some samples if they are full or leaving some empty space otherwise). A category mapping, unique for each reduction type, defines what category index a given sample belongs to[6].

When samples are combined in the reduction step, we can either subtract or add them together. Subtraction of samples is performed within a single category, while addition needs to take two samples from *adjacent* categories. For example, considering how plain-BKW reduction works for just one position with modulus $q$, there are $q$ different categories, one for each position value. Samples in categories $a$ and $q - a$ are adjacent since they cancel out (at that position) when added together. The category mapping is constructed so that *adjacent* categories are stored as neighbors on file, in order to maintain a sequential access pattern, avoiding the need for random accesses on file.

Consider one BKW reduction step. A reduction step takes a sample file as input and produces a new sample file as output. The fast dual SSDs of our target machine are used for efficiency here, reading from the source file, writing to the destination file. Sample reading is performed sequentially from beginning to end of file. This is very straightforward, reading a pair of adjacent categories into memory, processing them (the reduction), and writing the resulting samples to (another) file. Writing samples to the destination file is much more elaborate, utilizing as much of the RAM as is available as a buffer, flushing to disk with one sequential write to the destination file. This needs to be done whenever the buffer fills up, generally, but depending on the problem parameters, the amount of RAM and disk memory used, requiring multiple flushes.

### Optional Sample Amplification

We support optional sample amplification. That is, if a problem instance has a limited number of initial samples (e.g., the Darmstadt LWE challenge), then it is possible to combine several of these to produce new samples (more, but with higher noise).

While this is very straightforward in theory, we have noticed considerable performance effects when this recombination is performed naïvely. For example, combining

[6]In this section a category is defined slightly differently from the rest of the paper. A category together with its adjacent category are together what we simply refer to as a category in the rest of the paper.

triples of initial samples using a nested loop is problematic in practice for some instances, since some initial samples become over-represented – Some samples are used more often than others when implemented this way.

We have solved this by using a Linear Feedback Shift Register to efficiently and pseudo-randomly distribute the selection of initial samples more evenly.

#### Employing Meta-Categories

For some LWE problem instances, using a very high number of categories with few samples in each is a good option. This can be problematic to handle in an implementation, but we have used meta-categories to handle this situation. For example, using plain BKW reduction steps with modulus $q$ and three positions, we end up with $q^3$ different categories. With $q$ large, an option is to use only two out of the three position values in a vector to first map it into one out of $q^2$ different meta-categories. When processing the (meta-)categories, one then needs an additional pre-processing in form of a sorting step in order to divide the samples into their respective (non-meta) categories (based on all three position values), before proceeding as per usual.

We have used this implementation trick to, for example, implement plain BKW reduction for three positions. One may think of the process as brute-forcing one out of three positions in the reduction step.

#### Parallelization

The hybrid FWHT-based guessing method benefits from parallelization similarly to the hybrid guesser in the RBBL implementation. To every available thread we assign an equal number of possible guesses for the bruteforced part. Each thread independently runs an FWHT, and the most likely guess among all threads is chosen as the solution. Due to our focus on storage-related sample processing, we have not yet parallelized the BKW steps. This would require more effort compared to the RBBL case since one would want to parallelize the writing-to-file part too. It is in our future plan to work on it.

## 7.3 A Novel Idea for Fast Storage Writing

Here we introduce a new approach to handle file-based sample storage, which is particularly efficient for SSDs. That is, we describe a procedure for writing samples to physical

storage in an efficient way for large LWE instances with many samples. It is intended to be used in future implementations.

### Intuition

The main idea is to utilize the fact that disk access has much lower time penalty for SSD disks than for classical mechanical disks. In other words, we make use of the fact that SSDs act more like random access memory.

On a high level, the idea is to bunch several categories together into a well-chosen number of meta-categories. This can be done in a very general way, simply by grouping the category indices into intervals of suitable length (depending on available memory). We then make a rough sorting into these meta-categories, where we keep the samples unsorted (within their respective meta-category). For every such meta-category we utilise a separate file handle, so we have, say, $w$ different write positions on disk. The gain here is that we can simply employ the rough sorting and then directly flush each newly created sample (or batches of them if buffering is employed) into their respective file by appending. As mentioned, this technique works better for SSD-type disks than mechanical ones, because there is a penalty for physically moving the write head for every append operation. Of course, this can partially be dealt with by buffering the outputs, but this is a trade-off issue between performance gain of the algorithm itself versus the performance penalties of moving between the $w$ different write positions, which explains why the solution is more interesting for SSD storage.

### Technical description

Assume we have a table $T^{ram}$ in RAM and a larger table on disk denoted $T^{disk}$. Then, one splits the list of size $2^v$ into $m = 2^v/M$ different parts, where $M$ is related to the maximum RAM that can be used. Clearly, one such part matches the maximum RAM size, i.e., one part contains as many entries as can be contained in RAM.

Define a simple map $\varphi(K) = p$, where $p \in \{0, 1, \ldots, m-1\}$. A table $T^{ram}$ in RAM is created and contains $m$ storage units denoted $T_p^{ram}$, $p \in \{0, 1, \ldots, m-1\}$. When reading column $i$ from $\mathbf{A}$, one computes the corresponding index $K$ according to Equation (4.1) for the column $x$ and then the file part by $\varphi(K) = p$. The column vector $x$ is then stored in the storage units $T_p^{ram}$.

Once the table $T^{ram}$ is full, we append the parts to the larger table on disk $T^{disk}$, again containing $m$ storage units denoted $T_p^{disk}$, $p \in \{0, 1, \ldots, m-1\}$. Store the content

of $T_p^{ram}$ in $T_p^{disk}$, appended to previous content, for $p = 0, 1 \ldots, m-1$. Therefore, the table $T^{ram}$ is now empty and ready and start over again and read columns of $\mathbf{A}$.

In the combining step, we read the content of one $T_p^{disk}$ to RAM. But because the content is not fully sorted, we now address the table $T^{ram}$ by the $K$ values instead. Assume that $T^{ram}$ now is sorted according to $K$. Step through $T^{ram}$ and for all entries with the same $K$ value, we create all combinations of differences between these vectors. Write them to a new $\mathbf{A}$ in successive order.

## 7.4 Other Implementation Aspects

Some more things to consider implementation-wise are the following.

### Strict Unnatural Selection

There are advantages to performing very strict unnatural selection in the last reduction step, drastically reducing the total number of samples. First of all, this allows us to reduce more positions and/or reduce the positions to a much lower magnitude. Secondly, having much fewer samples speeds up the FWHT distinguisher, allowing us to brute-force guess more positions.

We leave investigating the idea of applying aggressive unnatural selection in each step, in other words coded-BKW with sieving, for future implementation work.

### Skipping the All 0s Guess When Using the FWHT Distinguisher

Consider the guess where all the LSBs of the secret are equal to 0. For that guess (5.5) simplifies to

$$\sum_{j=0}^{m-1} (-1)^{z''_j}. \tag{7.1}$$

Since even values of $z''_j$ are more common than odd values, this sum is biased, meaning that the FWHT is much more likely to choose this guess. Since it is exceptionally unlikely that this guess is the correct one, we can improve the distinguisher by simply discarding this guess.

### FWHT Distinguisher When the RAM is a Limitation

Suppose that the FWHT distinguisher is applied on $n$ positions, where the $2^n$ corresponding values are too many to store in RAM. It is possible to do a binary brute-force search over $r$ of the position values of (5.5) and calculate an FWHT with only $n-r$ bits for each of the possible $2^r$ bit sequences. This approach reduces the space complexity of the algorithm from $\mathcal{O}(2^n)$ to $\mathcal{O}(2^{n-r})$

The time complexity of the normal FWHT distinguisher, including the cost of processing the $m$ samples to calculate the $X_k$ values, is

$$\mathcal{O}(m + n \cdot 2^n). \tag{7.2}$$

When brute-forcing $r$ positions, we need to iterate over all $m$ samples and calculate a scalar product of $r$ positions, and calculate an FWHT of $n-r$ bits, for each of the $2^r$ guesses. This leads to a time complexity of

$$\mathcal{O}(2^r(rm + (n-r) \cdot 2^{n-r})). \tag{7.3}$$

### On the Minimum Population Size

It turns out that it is possible to use slightly less samples than a worst-case analysis would imply. The samples after a reduction step are not completely evenly spread out among the categories. The overproduction of categories with extra samples overcompensates for the lack of production from categories with few samples. The uneven spread of samples is due to two factors.

- By design, a small fraction of the categories will get fewer samples on average.
- Even ignoring the first point, the spread will due to randomness not be perfectly even.

Let us analyze the situation in detail. Assume that we have $N$ categories, $M$ samples and that the probability of a sample being mapped to category $i$ is $p_i$, where $\sum_{i=1}^{N} p_i = 1$. Let $X_i$ denote the number of samples in category $i$ and $Y_i = X_i(X_i - 1)/2$ denote the number of samples produces in category $i$. With this notation the expected new samples being produced is

$$\mathbb{E}\left(\sum_{i=1}^{N} Y_i\right) = \sum_{i=1}^{N} \frac{\mathbb{E}(X_i^2) - \mathbb{E}(X_i)}{2}.$$

The number of samples in category $i$ are $X_i \sim B(M, p_i)$. Using well-known formulas for the mean and the variance of the binomial distribution we get

$$\mathbb{E}(X_i^2) = V(X_i) + E(X_i)^2 = Mp_i(1-p_i) + M^2p_i^2,$$

leading to a total expected production of

$$\sum_{i=1}^{N} \frac{\mathbb{E}(X_i^2) - \mathbb{E}(X_i)}{2} = \sum_{i=1}^{N} \frac{Mp_i^2(M-1)}{2}.$$

Setting the production equal to $M$ and solving for $M$ gives us

$$M = 1 + \frac{2}{\sum_{i=1}^{N} p_i^2}$$

Notice here that further away from uniformly random the $p_i$ values are, the smaller we need $M$ to be. Assuming that all categories have $p_i = 1/N$ we get

$$M = 1 + 2N.$$

We can thus keep the sample size constant using $2N + 1$ samples, gaining a factor of $2/3$ over a worst-case analysis, which assumes that we need $3N$ samples. Notice that we gain from this effect even if we use larger values of $M$ than needed and choose the best samples using unnatural selection.

# 8 Experimental Results

In this section we report some of the experimental results obtained in solving real LWE instances with varying parameters. Our main goal was to confirm our theory and to prove that BKW algorithms can be used in practice to solve relatively large instances. For the case of FBBL, there is still room to run a more optimized code and possibly to make more optimal parameter choices. However, the results show that the BKW

algorithm is practical and that its performances are on a comparable scale to the ones from lattice-based approaches.

We considered two different scenarios. In the first case, we assumed for each LWE instance to have access to an arbitrary large number of samples. Here we create the desired amount of samples ourselves[7]. In the second case, we considered instances with a limited number of samples. An LWE problem is considered solved when the binary secret is correctly guessed, for reasons explained in Section 5.3.

### Target Machine

For our file-based experiments, we assembled a machine, that will be referred as *machine A*, to achieve a high speed in file reading/writing. We used an ASUS PRIME X399-A motherboard, a 4.0GHz Ryzen Threadripper 1950X processor and 128GiB of 2666MHz DDR4 RAM. For storage we used a separate (slow) SSD Samsung 860 QVO for the operating system (Windows 10), an Ultrastar HE12 12TB SATA mechanical disk, and dual (fast) SSDs SAMSUNG 970 EVO Plus 2TiB NVMe M.2 internal. While the machine is built from standard parts with a limited budget, we have primarily attempted to maximize the amount of RAM and the size and read/write speeds of the fast SSDs for overall ability to solve large LWE problem instances.

For the RAM-based experiments, we switched to a machine equipped with a faster processor. We used a desktop with processor 3.60GHz Intel Core i7-7700 CPU, running Linux Mint 20 and with 32 GB of RAM. We will refer to this second machine as *machine B*.

### Unlimited Number of Samples

We targeted the parameter choices of the TU Darmstadt challenges [7]. For each instance, we generated as many initial samples as needed according to our estimations. In Example 3 we present our parameter choices for one of these. In Table 8.1 we report the details of the largest solved instances. One can see that RBBL achieves considerably faster results than FBBL, both when comparing them on the same *machine B*, and when FBBL runs on *machine A*. This gives us an idea of how much the results obtained using FBBL on larger LWE instances could be improved if using RBBL on a machine with a larger RAM available.

**Example 3.** *Let us consider an LWE instance with* $n = 40$, $q = 1601$ *and* $\sigma = 0.005 \cdot q$. *To successfully guess the secret, we first performed 8 smooth-plain BKW steps reducing*

[7] we used rounded Gaussian noise for simplicity of implementation.

| $n$ | $q$ | $\alpha$ | number of initial samples | running time | library | machine | cores |
|---|---|---|---|---|---|---|---|
| 40 | 1601 | 0.005 | 16 M | 19 s | RBBL | B | 15 |
| 40 | 1601 | 0.005 | 16 M | 5 min 53 s | FBBL | B | 15 |
| 40 | 1601 | 0.005 | 16 M | 3 min 57 s | FBBL | A | 15 |
| 40 | 1601 | 0.010 | 570 M | 1 h 41 min | FBBL | A | 15 |
| 40 | 1601 | 0.015 | 4.2 B | 1 d 14 h | FBBL | A | 15 |
| 45 | 2027 | 0.005 | 250 M | 1 h 0 min | FBBL | A | 15 |
| 45 | 2027 | 0.010 | 8.3 B | 4 d 21.5 h | FBBL | A | 15 |
| 50 | 2503 | 0.005 | 2.7 B | 1 d 1.5 h | FBBL | A | 15 |

Table 8.1: Experimental results on target parameters. When using *FBBL*, parallelization was used only in the guessing phase, while the reduction phase was executed on a single core.

*17 positions to zero. We used the following parameters.*

$$\begin{aligned} n_i = 2, \quad C_i = 1, \qquad & \textit{for} \quad i = 1, \ldots, 8, \\ (C'_1, C'_2, C'_3, C'_4, C'_5, C'_6, C'_7, C'_8) = \quad & (280, 80, 20, 5, 1, 178, 41, 9). \end{aligned}$$

*Note that* $C'_5 = 1$. *In this way, we exploited the smoothness to zero one additional position. For this reason, we start step* 6 *by skipping one position. Finally, we did 5 smooth-LMS steps using the following parameters:*

$$\begin{aligned} (n_9, n_{10}, n_{11}, n_{12}, n_{13}) &= (4, 4, 4, 5, 5) \\ (C_9, C_{10}, C_{11}, C_{12}, C_{13}) &= (13, 24, 33, 48, 66) \\ (C'_9, C'_{10}, C'_{11}, C'_{12}, C'_{13}) &= (267, 534, 321, 48, 66). \end{aligned}$$

*These parameters are chosen in such a way that the number of categories*[8] *is* $\approx 15M$ *in the early stages and* $\approx 23M$ *at its most. We started with* $16M$ *samples that guaranteed us to end up with enough samples for guessing the right solution. The last position is brute-forced and therefore left untouched at the last reduction step.*

### Limited Number of Samples

We solved the original TU Darmstadt LWE challenge instance [7] with parameters $n = 40$, $\alpha = 0.005$ and the number of samples limited to $m = 1600$. We did this by forming 140 million samples using sample amplifying with triples of samples, taking 6 steps of smooth-plain BKW on 14 entries, followed by 6 steps of smooth-LMS on 25 entries. The

[8] The number of categories here is the double of what is explained in previous sections since opposite samples are put in different categories in the implementation.

final position was left to brute-force. The overall running time, obtained with FBBL on *machine A*, was 55 minutes.

# 9 Conclusions and Future Work

We introduced a novel and easy approach to implementing a BKW reduction step, which allows balancing the complexity among the iterations, and an FWHT-based guessing procedure able to correctly guess the secret with relatively large noise level. Together with a file-based approach of storing samples, the above define a new BKW algorithm specifically designed to solve practical LWE instances, where the available RAM is typically a limiting factor.

With an implementation of the file-based algorithm, we managed to solve 6 challenges with Darmstadt Challenge-type parameters, but with unlimited number of samples, 3 more challenges than in the conference version of the paper [39]. For the 3 previously solved challenges, we made substantial improvements in runtime. We also managed to solve the easiest Darmstadt challenge, in its original form.

Furthermore, we implemented a fully RAM-based version of the new algorithm, to compare against the file-based approach, in settings where the available RAM was not a limiting factor. We leave to future work also experimenting such implementation with harder LWE instances on machines with larger RAM available. We did parallelize the FWHT and the RAM-based version of the algorithm, but we leave more parallelization work and other optimization work for the future.

While we managed to substantially improve the implementation results of the conference version of the paper [39], we believe that significant improvements to the algorithm can still be made to reduce the gap compared to lattice-based techniques for solving LWE. For example, it remains to investigate the concrete improvement of employing the sieving aspect of *Coded-BKW with Sieving* [22]. Also, the investigation of the specific design of the BKW algorithm for handling the problem of few initial samples is left for future work.

# Bibliography

[1] P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[2] "NIST Post-Quantum Cryptography Standardization."

https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization. Accessed: 2018-09-24.

[3] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pp. 84–93, ACM, 2005.

[4] A. Blum, M. Furst, M. Kearns, and R. J. Lipton, "Cryptographic Primitives Based on Hard Learning Problems," in *Advances in Cryptology — CRYPTO' 93*, vol. 773 of *LNCS*, pp. 278–291, Springer, 1994.

[5] S. Arora and R. Ge, "New Algorithms for Learning in Presence of Errors," in *Automata, Languages and Programming*, vol. 6755 of *LNCS*, pp. 403–415, Springer, 2011.

[6] M. R. Albrecht, C. Cid, J. C. Faugère, R. Fitzpatrick, and L. Perret, "Algebraic Algorithms for LWE Problems," in *ACM Communications in Computer Algebra*, vol. 49, pp. 62–91, ACM, 2015.

[7] "TU Darmstadt Learning with Errors Challenge." https://www.latticechallenge.org/lwe_challenge/challenge.php. Accessed: 2020-05-01.

[8] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The General Sieve Kernel and New Records in Lattice Reduction," in *Advances in Cryptology – EUROCRYPT 2019*, vol. 11477 of *LNCS*, pp. 717–746, Springer, 2019.

[9] M. R. Albrecht, R. Player, and S. Scott, "On the Concrete Hardness of Learning With Errors," in *Journal of Mathematical Cryptology*, vol. 9, De Gruyter, 2015.

[10] G. Herold, E. Kirshanova, and A. May, "On the Asymptotic Complexity of Solving LWE," *Designs, Codes and Cryptography*, vol. 86, pp. 55–83, 2018.

[11] Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski Wagner, "On the Asymptotics of Solving the LWE Problem Using Coded-BKW with Sieving," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 5243–5259, 2019.

[12] S. Katsumata, K. Kwiatkowski, F. Pintore, and T. Prest, "Scalable Ciphertext Compression Techniques for Post-quantum KEMs and Their Applications," in *Advances in Cryptology – ASIACRYPT 2020*, vol. 12491 of *LNCS*, pp. 289–320, Springer, 2020.

[13] A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant learning, the parity problem, and the statistical query model," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pp. 435–440, ACM, 2000.

[14] É. Levieil and P.-A. Fouque, "An Improved LPN Algorithm," in *Security and Cryptography for Networks*, vol. 4116 of *LNCS*, pp. 348–359, Springer, 2006.

[15] Q. Guo, T. Johansson, and C. Löndahl, "Solving LPN Using Covering Codes," in *Advances in Cryptology – ASIACRYPT 2014*, vol. 8873 of *LNCS*, pp. 1–20, Springer, 2014.

[16] Q. Guo, T. Johansson, and C. Löndahl, "Solving LPN Using Covering Codes," *Journal of Cryptology*, vol. 33, no. 1, pp. 1–33, 2020.

[17] M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, and L. Perret, "On the Complexity of the BKW Algorithm on LWE," *Design, Codes and Cryptography*, vol. 74, no. 2, pp. 325–354, 2015.

[18] M. R. Albrecht, J.-C. Faugère, R. Fitzpatrick, and L. Perret, "Lazy Modulus Switching for the BKW Algorithm on LWE," in *Proceedings of the 17th International Conference on Public-Key Cryptography – PKC 2014*, vol. 8383 of *LNCS*, pp. 429–445, Springer, 2014.

[19] Q. Guo, T. Johansson, and P. Stankovski, "Coded-BKW: Solving LWE Using Lattice Codes," in *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *LNCS*, pp. 23–42, Springer, 2015.

[20] P. Kirchner and P. Fouque, "An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices," in *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *LNCS*, pp. 43–62, Springer, 2015.

[21] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, "New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 10–24, SIAM, 2016.

[22] Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski, "Coded-BKW with Sieving," in *Advances in Cryptology – ASIACRYPT 2017*, vol. 10624 of *LNCS*, pp. 323–346, Springer, 2017.

[23] E. Mårtensson, "The Asymptotic Complexity of Coded-BKW with Sieving Using Increasing Reduction Factors," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 2579–2583, IEEE, 2019.

[24] A. Duc, F. Tramèr, and S. Vaudenay, "Better Algorithms for LWE and LWR," in *Advances in Cryptology – EUROCRYPT 2015*, vol. 9056 of *LNCS*, pp. 173–202, Springer, 2015.

[25] A. Esser, R. Kübler, and A. May, "LPN Decoded," in *Advances in Cryptology – CRYPTO 2017*, vol. 10402 of *LNCS*, pp. 486–514, Springer, 2017.

[26] A. Esser, F. Heuer, , R. Kübler, A. May, and C. Sohler, "Dissection-BKW," in *Advances in Cryptology – CRYPTO 2018*, vol. 10992 of *LNCS*, pp. 638–666, Springer, 2018.

[27] C. Delaplace, A. Esser, and A. May, "Improved Low-Memory Subset Sum and LPN Algorithms via Multiple Collisions," in *Cryptography and Coding - 17th IMA International Conference*, vol. 11929 of *LNCS*, pp. 178–199, Springer, 2019.

[28] T. Wiggers and S. Samardjiska, "Practically Solving LPN," in *2021 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2021.

[29] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems," in *Advances in Cryptology - CRYPTO 2009*, vol. 5677 of *LNCS*, pp. 595–618, Springer, 2009.

[30] P. Kirchner, "Improved Generalized Birthday Attack." Cryptology ePrint Archive, Report 2011/377, 2011. https://ia.cr/2011/377.

[31] R. Lindner and C. Peikert, "Better Key Sizes (and Attacks) for LWE-Based Encryption," in *Topics in Cryptology – CT-RSA 2011*, vol. 6558 of *LNCS*, pp. 319–339, Springer, 2011.

[32] E. D. Mulder, M. Hutter, M. E. Marson, and P. Pearson, "Using Bleichenbacher's Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-bit ECDSA: Extended Version," *Journal of Cryptographic Engineering*, vol. 4, no. 1, pp. 33–45, 2014.

[33] Q. Guo, E. Mårtensson, and P. Stankovski Wagner, "On the Sample Complexity of solving LWE using BKW-Style Algorithms," in *2021 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2021.

[34] T. Baignères, P. Junod, and S. Vaudenay, "How Far Can We Go Beyond Linear Cryptanalysis?," in *Advances in Cryptology - ASIACRYPT 2004*, vol. 3329 of *LNCS*, pp. 432–450, Springer, 2004.

[35] E. Mårtensson, *Some Notes on Post-Quantum Cryptanalysis*. PhD thesis, Lund University, 2020. https://lup.lub.lu.se/search/files/88158602/thesis_main_document.pdf.

[36] W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers," *Journal of Cryptology*, vol. 1, pp. 159–176, 1989.

[37] P. Chose, A. Joux, and M. Mitton, "Fast Correlation Attacks: An Algorithmic Point of View," in *Advances in Cryptology — EUROCRYPT 2002*, vol. 9215 of *LNCS*, pp. 209–221, Springer, 2002.

[38] Y. Lu, W. Meier, and S. Vaudenay, "The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption," in *Advances in Cryptology – CRYPTO 2005*, pp. 97–117, Springer, 2005.

[39] A. Budroni, Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski Wagner, "Making the BKW Algorithm Practical for LWE," in *Progress in Cryptology – INDOCRYPT 2020*, vol. 12578 of *LNCS*, pp. 417–439, Springer, 2020.

# Paper 4

# Efficient Hash Maps to $\mathbb{G}_2$ on BLS curves

*Alessandro Budroni and Federico Pintore*

**Abstract:** When a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, on an elliptic curve $E$ defined over a finite field $\mathbb{F}_q$, is exploited for an identity-based protocol, there is often the need to hash binary strings into $\mathbb{G}_1$ and $\mathbb{G}_2$. Traditionally, $\mathbb{G}_1 = E(\mathbb{F}_q) \cap E[r]$, where $r$ is a prime integer, and $\mathbb{G}_2 = \tilde{E}(\mathbb{F}_{q^{k/d}}) \cap \tilde{E}[r]$, where $\tilde{E}$ is a twist of $E$ of order $d$, and $k$ is the embedding degree of $E$ w.r.t. $r$. The standard approach for hashing into $\mathbb{G}_2$ is to map to a general point $P \in \tilde{E}(\mathbb{F}_{q^{k/d}})$ and then multiply it by the cofactor $c = \#\tilde{E}(\mathbb{F}_{q^{k/d}})/r$. Usually, the multiplication by $c$ is computationally expensive. In order to speed up such a computation, two different methods - by Scott *et al.* [1] and by Fuentes *et al.* [2] - have been proposed. In this paper we consider these two methods for BLS pairing-friendly curves having $k \in \{12, 24, 30, 42, 48\}$, providing efficiency comparisons. When $k = 42, 48$, the application of Fuentes *et al.* method requires expensive computations which were infeasible for the computational power at our disposal. For these cases, we propose hashing maps that we obtained following Fuentes *et al.* idea.

**Keywords:** Pairing-based Cryptography, Pairing-friendly Elliptic Curves, Fast Hashing.

## 1 Introduction

### 1.1 Pairings in Cryptography

Pairings on elliptic curves have been first used in Cryptography to transport elliptic curve discrete logarithms into finite field discrete logarithms ([3], [4]), for which there are index-calculus algorithms running in subexponential time. In recent years, several protocols have been proposed with pairings on elliptic curves as building blocks. Among them, it is possible to enumerate Joux's three party key agreement protocol [5], a non-interactive key-exchange [6], an identity-based encryption [7], and a short signatures scheme [8].

Traditionally, pairings that have been considered for applications are the Tate and Weil pairings on elliptic curves over finite fields, and other related pairings, for example the Eta pairing [9], the Ate pairing [10], and their generalisations [11]. Let $\mathbb{F}_q$ be a finite field with characteristic $p \neq 2, 3$ and order $q = p^m$, for some $m > 0$. For an elliptic curve $E$ defined over $\mathbb{F}_q$, all these pairings take as inputs points on $E(\mathbb{F}_q)$ or on $E(\mathbb{F}_{q^k})$ - where $\mathbb{F}_{q^k}$ is an extension field of the base field $\mathbb{F}_q$ - and return as outputs elements of $(\mathbb{F}_{q^k})^*$.

In this paper we will only consider asymmetric pairings $e$. In particular, given a prime $r$ such that $r||\#E(\mathbb{F}_q)$ (i.e. $r|\#E(\mathbb{F}_q)$ but $r^2 \nmid \#E(\mathbb{F}_q)$), then $e$ will be of the form:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

where $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve subgroups of order $r$ defined as:

- $\mathbb{G}_1 = E(\mathbb{F}_q) \cap E[r]$,
- $\mathbb{G}_2 = E[r] \cap \ker(\pi - [q])$, where $\pi$ is the *Frobenius endomorphism* and $[q] : E(\mathbb{F}_q) \to E(\mathbb{F}_q)$ maps $P \mapsto qP$,

while $\mathbb{G}_T$ is a subgroup of order $r$ of $(\mathbb{F}_{q^k})^*$. With $k$ is denoted the embedding degree of $E$ with respect to $r$, i.e. the smallest positive integer such that $r \mid q^k - 1$. Such $\mathbb{G}_2$ always exists as the Frobenius endomorphism on $E[r]$ has eigenvalues 1 and $q$.

For pairing-based cryptosystems to be secure, the discrete logarithm problem on both $E(\mathbb{F}_q)$ and $(\mathbb{F}_{q^k})^*$ must be computationally infeasible. The elliptic curves satisfying this condition, for a relatively small embedding degree $k$, are called *pairing-friendly elliptic curves*. The first formal definition of pairing-friendly elliptic curves has been formulated by Freeman *et al.* in their comprehensive paper [12].

## 1.2 Families of pairing-friendly elliptic curves

The works of Balasubramanian and Koblitz [13] and Luca *et al.* [14] show that pairing-friendly elliptic curves are rare, and hence they require dedicated constructions. In recent years a number of methods for constructing such curves have been proposed [15], [16], [17], [18], [19], [20]. The general pattern is the same for all of them: given an embedding degree $k$, find three integers $n, r, q$ for which there exists an elliptic curve $E$ defined over $\mathbb{F}_q$ and such that

- $\#E(\mathbb{F}_q) = n$,

- $r \geq \sqrt{q}$ is a prime factor of $n$,
- $k$ is the embedding degree of $E$ w.r.t. $r$.

Then the complex multiplication (CM) method [21] is used to determine the equation of the above elliptic curve $E$.

However, instead of producing single pairing-friendly elliptic curves by means of specific integers $k, n, r, q$, all the cited methods produce *families* of pairing-friendly elliptic curves. In particular, the integers $n, r, q$ are replaced by suitable polynomials $n(x)$, $r(x)$, $q(x) \in \mathbb{Q}[x]$. For some appropriate $x_0 \in \mathbb{Z}$, $n(x_0), r(x_0), q(x_0)$ are integers such that there exists an elliptic curve $E$ defined over $\mathbb{F}_{q(x_0)}$, having $n(x_0)$ rational points, with prime $r(x_0)||n(x_0)$, and $k$ as embedding degree w.r.t. $r(x_0)$. The triple $(n(x), r(x), q(x))$ defines a *family* of pairing-friendly elliptic curves, each of them parametrised by the integers $n(x_0), r(x_0), q(x_0)$ for some $x_0 \in \mathbb{Z}$. If for every $x_0 \in \mathbb{Z}$ there exists an elliptic curve with $n(x_0), r(x_0), q(x_0)$ as parameters, the family defined by $(n(x), r(x), q(x))$ is said to be *complete*, otherwise it is called *sparse*.

The paring-friendly (sparse or complete) families of curves obtained with the methods enumerated above are known as MNT curves [15], BLS curves [16], [17], BN curves [18], Freeman curves [19] and KSS curves [20], respectively.

## 1.3 Hashing to $\mathbb{G}_2$

When pairings on elliptic curves are used for identity-based protocols, there is often a need to map binary strings representing the identity of the parties into random points of $\mathbb{G}_1$ and $\mathbb{G}_2$. These tasks are known as *hashing to* $\mathbb{G}_1$ and *hashing to* $\mathbb{G}_2$ respectively.

Hashing to $\mathbb{G}_1$ is relatively easy. In fact, since $\mathbb{G}_1$ is the unique subgroup of order $r$ in $E(\mathbb{F}_q)$ (thanks to the assumption $r||\#E(\mathbb{F}_q)$), the standard approach is to hash to a point $P \in E(\mathbb{F}_q)$ and then multiply it by the cofactor $c = \#E(\mathbb{F}_q)/r$. On the other hand, if $E$ admits a twist[1] $\tilde{E}$ of degree $d$ that divides $k$, then $\mathbb{G}_2$ is isomorphic to $\tilde{E}(\mathbb{F}_{q^{k/d}}) \cap \tilde{E}[r]$.Consequently the same approach can be used for hashing into $\mathbb{G}_2$. Nevertheless, the latter requires a multiplication by a large cofactor and hence expensive computations.

We note that the intermediate step of hashing into a general rational point should be handled carefully for efficiency and security reasons. In particular, some cryptosystems are proved to be secure when such an intermediate hash function is modelled as a random

[1] an elliptic curve isomorphic to $E$ over an extension field of $\mathbb{F}_q$.

oracle into the curve. In order to guarantee its secure replacement with the random oracle, the concept of *indifferentiable* hash function has been introduced [22].

## 1.4 Related Work

In 2009, Scott *et al.* [1] exploited an efficiently-computable endomorphism $\psi : \tilde{E} \to \tilde{E}$ to reduce the computational cost of the cofactor multiplication required for hashing to $\mathbb{G}_2$. An improvement of this method was then proposed by Fuentes *et al.* in 2011 [2]. Since pairing-friendly families vary significantly, in order to highlight the benefits of the two methods, families of curves were considered case-by-case in [1] and in [2]. In particular, both papers focus on BN curves with $k = 12$, Freeman curves with $k = 10$ and KSS curves with $k = 8, 18$. However, new advances on the Number Field Sieve (NFS) ([23], [24], [25]) for computing discrete logarithms in multiplicative groups of finite fields, and hence in $\mathbb{G}_T$, have decreased the security of some asymmetric parings, including those built on BN curves [26], [27]. In the light of these results, BLS curves are attracting more interest for efficiency reasons, since their security has been only slightly reduced by recent NFS advances [26], [27].

Scott *et al.* and Fuentes *et al.* methods are the only two proposed so far that improve upon standard cofactor multiplication for hashing to $\mathbb{G}_2$. However, to the best of our knowledge, there are not published sources explicitly applying both Scott *et al.* and Fuentes *et al.* methods to BLS curves with $k \in \{12, 24, 30, 42, 48\}$, and providing efficiency comparison of the outcomes.

## 1.5 Contributions and Outline

In this paper that gap is filled for BLS curves with $k = 12, 24, 30$. We provide a comparison on the efficiency of the two methods. In a recently-published book [28], it is stated that, for BLS curves with $k = 12, 24$, the most efficient method for mapping into $\mathbb{G}_2$ is the one proposed by Scott *et al.* This is opposite to our results.

Both methods require a pre-computation to obtain parameterised hashing formulas valid for all the curves that belong to a specific family of pairing-friendly curves. In particular, Scott *et al.* method needs polynomial modular arithmetic, while Fuentes *et al.* method requires the application of a generalized LLL algorithm to a polynomial matrix. In this paper, we first applied Scott *et al.* method to BLS curves with $k \in \{12, 24, 30, 42, 48\}$. Then we explicitly applied Fuentes *et al.* method to BLS curves with $k \in \{12, 24, 30\}$. However, running LLL for the other two remaining cases revealed

to be infeasible for our resources. Nevertheless, for $k \in \{42, 48\}$, we propose suitable hash maps which allow to speed up the execution of cofactor multiplications with respect to Scott *et al.* method.

Our efficiency conclusions are that hashing on BLS curves following Fuentes *et al.* method is faster than applying Scott *et al.* method, for every $k \in \{12, 24, 30, 42, 48\}$.

The remainder of this paper is organized as follows. In Section 2 we recall Scott *et al.* and Fuentes *et al.* methods. For the sake of easy reference, in Subsection 2.1 we summarise BLS curves' parameters. In Section 3, the Scott *et al.* method is applied to BLS curves with embedding degree $k \in \{12, 24, 30, 42, 48\}$. In Section 4, Fuentes *et al.* method is applied to BLS curves with $k \in \{12, 24, 30\}$. In Section 5, we exploit some patterns in the maps computed with Fuentes *et al.* method to retrieve hash maps for BLS curves with $k = 42, 48$ which are more efficient than the ones presented in Section 3. Finally, an efficiency comparison of the maps for BLS curves is provided in Section 6.

# 2 Known methods for efficiently mapping into $\mathbb{G}_2$

The problem of generating random points in $\mathbb{G}_2$, known as *hashing to* $\mathbb{G}_2$, is usually solved selecting a random point $P \in \tilde{E}(\mathbb{F}_{q^{k/d}})$ and then computing $[c]P$, where $c$ is the cofactor defined as $c = \#\tilde{E}(\mathbb{F}_{q^{k/d}})/r$. Due to the size of $c$, this scalar multiplication is generally expensive and consequently a bottleneck in hashing to $\mathbb{G}_2$.

In [29], Gallant, Lambert and Vanstone give a method to speed up scalar multiplications $[w]P$ in $E(\mathbb{F}_q)[r]$. This method is based on the knowledge of a non-trivial multiple of the point $P$, that is obtained from an efficiently computable endomorphism $\omega : E \to E$ such that $\omega(P)$ is a multiple of $P$.

## 2.1 Scott *et al.* method

Building on the above idea, Galbraith and Scott [30, Section 8] reduced the computational cost of multiplying by the cofactor $c$ for BN curves by using a suitable group endomorphism $\psi : \tilde{E} \to \tilde{E}$. This method was later improved and generalized by Scott *et al.* [1]. Such an endomorphism is defined as $\psi = \varphi^{-1} \circ \pi \circ \varphi$, where $\pi$ is the $q$-power Frobenius on $E$ and $\varphi$ is an isomorphism from the twist curve $\tilde{E}$ to $E$. The endomorphism $\psi$ satisfies

$$\psi^2(P) - [t]\psi(P) + [p]P = \infty \tag{2.1}$$

for all $P \in \tilde{E}(\mathbb{F}_{p^2})$. In the above relation, $t$ is the trace of Frobenius $p+1-\#E(\mathbb{F}_p)$. Given that the cofactor of $\mathbb{G}_2$ for BN curves is $c = p-1+t$, with a simple substitution, one gets

$$[c]P = [p-1+t]P = [t](\pi(P)+P) - \pi^2(P) - P.$$

This gives us a speed up as $t$ is about half the size of $c$, yielding to a less expensive scalar multiplication. The generalization of the method introduced by Scott *et al.* works as follows. Express first the cofactor $c$ to the base $p$ as

$$c = c_0 + c_1 p + \cdots + c_\ell p^\ell. \tag{2.2}$$

Then use (2.1) to simplify the multiplication $[c]P$ as

$$[c]P = [c_0 + c_1 p + \cdots + c_\ell p^\ell]P = [g_0]P + [g_1]\psi(P) + \cdots + [g_{2\ell}]\psi^{2\ell}(P) \tag{2.3}$$

for some $\mid g_i \mid < p$ for every $i$.

This approach was applied to several families of pairing-friendly curves. In particular, the curves taken into account in [1] are: the MNT curves for the case $k = 6$, the BN curves with $k = 12$, the Freeman curves with $k = 10$ and the KSS curves for the cases $k = 8$ and $k = 18$. It is important to highlight that all these families are composed by curves defined over a prime field $\mathbb{F}_p$, with $p$, the order $r$ and the trace $t$ expressed as polynomials having rational coefficients. Consequently, also the cofactor $c$ can be described as a polynomial in $\mathbb{Q}[x]$. Thanks to such a parameterisation, Scott *et al.* speed up the cofactor multiplication $[c]P$ reducing it to the evaluation of a polynomial of the powers $\psi^i(P)$, with coefficients that are polynomials in $x$. Such coefficients are obtained by means of polynomial modular arithmetic. In particular, all these coefficients have degrees smaller than $\deg(p(x))$ (for the same reason, numerical coefficients $g_i$ are bounded by $q$).

## 2.2 Fuentes *et al.* method

Fuentes *et al.* [2] improved Scott *et al.* method observing that, in order to obtain a non-zero multiple of $P \in \tilde{E}(\mathbb{F}_{q^{k/d}})$ having order $r$, it is sufficient to multiply $P$ by $c'$, a multiple of $c$ such that $c' \not\equiv 0 \pmod{r}$. In particular they proved the following result (see [2], page 11):

**Theorem 1.** *If $\tilde{E}(\mathbb{F}_{q^{k/d}})$ is cyclic and $q \equiv 1 \pmod{d}$, then there exists a polynomial*

$$h(z) = h_0 + h_1 z + \cdots + h_{\varphi(k)-1} z^{\varphi(k)-1} \in \mathbb{Z}[z] \tag{2.4}$$

*such that:*

- *$h(\psi)P$ is a multiple of $[c]P$ for all $P \in \tilde{E}(\mathbb{F}_{q^{k/d}})$;*
- *the coefficients of $h(z)$ satisfy $\mid h_i \mid^{\varphi(k)} \leq c$ for all $i$.*

*We note that here $\varphi$ stands for the Euler's totient function, while $\psi$ is the efficiently computable endomorphism satisfying (2.1).*

The first condition about $h(z)$ gives a tool for computing a multiple of $[c]P$ as the sum of some scalar multiplications. These multiplications are computationally light since their scalar factors are bounded thanks to the second condition satisfied by $h(z)$.

The proof of Thereom 1 is by construction, exploiting the *LLL algorithm* of Lenstra, Lenstra and Lovász [31]. For the sake of easy reference we sketch the proof. We refer the reader to [28, Sec. 8.5] and [2, Sec. 5] for more details.

Let $q = p^{k/d}$ and let $\hat{t}$ be the trace of the $q-$power Frobenius of $E$. This can be computed from $t$ using the recursive formula $t_0 = 2$, $t_1 = t$, and $t_{i+1} = t \cdot t_i - p \cdot t_{i-1}$ [32]. Then, we set $\hat{t} = t_{k/d}$. We have that $\#\tilde{E}(\mathbb{F}_q) = q + 1 + \hat{t}$ (see [10]), and let $\hat{f}$ be such that $\hat{t}^2 - 4q = D\hat{f}^2$, where $D$ is square-free.
The trace $\tilde{t}$ of the $q$-power of $\tilde{E}$ over $\mathbb{F}_q$, for $d = 2, 3, 4, 6$ can be computed using the formulas given by Hess *et al.* [10]. For the case of our interest $d = 6$, $\tilde{t}$ is equal to $(\pm 3\hat{f} + \hat{t})/2$.
Let $\tilde{n}$ denote the cardinality $\#\tilde{E}(\mathbb{F}_q) = q + 1 + \tilde{t}$, and let $\tilde{f}$ be the integer such that $\tilde{t}^2 - 4q = D\tilde{f}^2$. Analogously, we denote with $f$ the integer for which $t^2 - 4p = Df^2$ holds.

First of all, it is observed ([2, Lemma 2]) that if $\gcd(\tilde{n}, \tilde{f}) = 1$, then, for every point $P \in \tilde{E}(\mathbb{F}_q)$, one has that $\psi(P) = [a]P$, where:

$$a = \frac{t}{2} + \frac{f(\tilde{t} - 2)}{2\tilde{f}} \mod \tilde{n} \quad \text{or} \quad a = \frac{t}{2} - \frac{f(\tilde{t} - 2)}{2\tilde{f}} \mod \tilde{n}, \tag{2.5}$$

and therefore $h(\psi)P = [h(a)]P$ . Then, the relation

$$\psi^k_{|\tilde{E}(\mathbb{F}_q)} = id_{\tilde{E}(\mathbb{F}_q)},$$

where $\psi_{|\tilde{E}(\mathbb{F}_q)}$ denotes a restriction of $\psi$ on $\tilde{E}(\mathbb{F}_q)$, is obtained. Hence $\Phi_k(a) \equiv 0 \pmod{\tilde{n}}$, where $\Phi_k$ is the $k$-th cyclotomic polynomial (which has degree equal to $\varphi(k)$). This allows to restrict the search of $h(z)$ into the set of all polynomials of $\mathbb{Z}[z]$ of degree less than $\varphi(k)$. Let $\mathbf{a}$ denote the column vector with $i$-entry $-a^i$. Consider the vectors of the

integer lattice generated by the matrix

$$M = \left[\begin{array}{c|c} c & \mathbf{0} \\ \hline \mathbf{a} & I_{\varphi(k)-1} \end{array}\right].$$

This lattice is of dimension $\varphi(k)$ and volume $c$. Any vector $(h_0, h_1, ..., h_{\varphi(k)-1})$ of the lattice results in a polynomial $h(z) = h_0 + h_1 z + ... + h_{\varphi(k)-1} z^{\varphi(k)-1} \in \mathbb{Z}[z]$ such that $h(a) \equiv 0 \pmod{c}$. Finally, it is observed that the considered lattice and the convex set generated by all vectors of the form $(\pm \mid c \mid^{1/\varphi(k)}, \ldots, \pm \mid c \mid^{1/\varphi(k)})$ have non-empty intersection. Such an element lying in this intersection could be obtained using the *LLL algorithm*[2] [31] and determines the coefficients of a polynomial $h(z) \in \mathbb{Z}[z]$ with the desired properties.

In [2], such a polynomial is obtained for the BN curves with $k = 12$, the Freeman curves with $k = 10$, the KSS curves for the cases $k = 8$ and $k = 18$. As already observed, these families are composed by curves defined over a prime field $\mathbb{F}_p$, with $p$, $q$, $r$, $t$, $f$, $\hat{t}$, $\hat{f}$, $\tilde{t}$, $\tilde{f}$ and $\tilde{n}$ expressed as polynomials with rational coefficients. For example, for an elliptic curve $E$ with complex multiplication, one computes the polynomials $f(x)$ and $\tilde{f}(x)$ such that $t^2(x) - 4p(x) = Df^2(x)$ and $\tilde{t}^2(x) - 4q(x) = D\tilde{f}^2(x)$, where $D$ is a square free integer. Consequently, using the same formulas as above, but using polynomials at the place of scalars ($r(x)$ instead of $r$, and so on), also the cofactor $c$ can be described as a polynomials $c(x) \in \mathbb{Q}[x]$, and one obtains a parametrization of $a$ in (2.5) by a rational function

$$a(x) = \frac{t(x)}{2} \pm \frac{f(x)(\tilde{t}(x) - 2)}{2\tilde{f}(x)} \mod \tilde{n}(x). \tag{2.6}$$

The matrix $M$ for the parameterised $c$ and $a$ is

$$M = \left[\begin{array}{c|c} c(x) & \mathbf{0} \\ \hline \mathbf{a}(x) & I_{\varphi(k)-1} \end{array}\right],$$

where $\mathbf{a}$(x) is the column vector with $i$-entry $-a^i(x) \pmod{c(x)}$, and it generates a lattice in $\mathbb{Q}[x]^{\varphi(k)}$. The algorithm in [33], takes $M$ as input and returns a matrix $M'$ having as rows a reduced basis for the lattice generated by $M$. Considering the polynomials that compose a row of $M'$ as coefficients of 1, $z$, $z^2$, ..., $z^{\varphi(k)-1}$ respectively, Fuentes *et al.* were able to obtain a polynomial $h(z) = \sum_i h_i(x) z^i \in \mathbb{Z}[x][z]$ which satisfy the following two conditions:

(CI) $h(a(x)) \equiv s(x)c(x) \pmod{\tilde{n}(x)}$, with $\gcd(s(x), r(x)) = 1$, for some $s(x) \in \mathbb{Q}[x]$;

[2]Not that this holds only for relatively small $k$

(CII) $\deg(h_i(x)) \leq \deg(c(x))/\varphi(k)$, where $\varphi$ is the Euler's totient function.

The first condition assures that $[h(a(x_0))]P$ is a non-zero multiple of $[c(x_0)]P$ for every value $x_0 \in \mathbb{Z}$ of the parameter $x$, and that such a multiple can be computed as the sum of some scalar multiplications. These multiplications are computationally light thanks to the second condition in which scalar factors are bounded.

Consequently, for the BN, Freeman and KSS pairing-friendly families curves, Fuentes *et al.* compute a formula for hashing into $\mathbb{G}_2$ that is valid for every curve in the family itself. In particular, the cofactor multiplication $[c(x)]P$ is reduced to the evaluation of a polynomial of the powers $\psi^i(P)$, with coefficients that are polynomials in $x$. Fuentes *et al.* provided evidence that their method is faster than that of Scott *et al.* for the same curves from the families BN, Freeman and KSS.

## 2.3 BLS curves

Families of pairing-friendly curves vary significantly, hence it is not possible to a priori determine if one of the two above hashing methods is more efficient than the other for a given family. BLS curves recently gained interest [26], [27]. Thus it is of interest to determine also for these curves which is, among Scott *et al.* and Fuentes *et al.* methods, the more efficient one. In [28, Section 8.5], the Scott *et al.* method is explicitly applied to BLS curves with $k \in \{12, 24\}$ and it is stated that, for these cases, this produces the most efficient hash maps.

In this paper we derive formulas for BLS curves with $k = \{12, 24, 30\}$ using both methods. We provide evidences that, on the contrary, the most efficient method is the one of Fuentes *et al.* Furthermore, we apply Scott *et al.* method also to BLS curves with $k \in \{42, 48\}$. The computations necessary, within Fuentes *et al.* method, to obtain the polynomial $h(z)$ for BLS curves with $k = 42, 48$ were infeasible for the computational power at our disposal. In Section 5, we propose polynomials $\mathfrak{h}(z) = \sum_i h(x)z^i$ which satisfy (CI). That is $\mathfrak{h}(a(x))$ is congruent to a multiple of $c(x)$ modulo $\tilde{n}(x)$. For $k = 48$ the polynomial satisfy $\deg(\mathfrak{h}_i(x)) \leq \deg(c(x))/\varphi(k)$ for every $i$, satisfying therefore condition (CII) too. For $k = 42$, the same condition is satisfied for every $\mathfrak{h}_i(x)$ except for $\mathfrak{h}_0(x)$ that has degree equal to $\lfloor \deg(c(x))/\varphi(k) \rfloor + 1$.

We conclude this section briefly recalling BLS curves' parameters. Barreto, Lynn and Scott [16], and Brezing and Weng [17] proposed a polynomial parameterisation for complete families of pairing-friendly curves having prime fields $\mathbb{F}_p$ as basefields, fixed embedding degrees, and short Weierstrass equations of the form $y^2 = x^3 + b$.

In the following, we consider only those BLS curves with embedding degree $k = 12, 24, 30, 42, 48$. For all cases, we have that $k$ is a multiple of 6. This choice is due to efficiency reasons, since every such curve admits a twist of the highest possible degree $d = 6$ [10], allowing to consider $\mathbb{G}_2$ as a subgroup of $\tilde{E}(\mathbb{F}_{p^{k/6}})$. In this case BLS curves are parameterised by the following polynomials [12]:

$$\begin{aligned} p(x) &= \frac{1}{3}(x-1)^2(x^{k/3} - x^{k/6} + 1) + x, \\ r(x) &= \Phi_k(x), \\ t(x) &= x + 1, \end{aligned}$$

where $\Phi_k$ is the cyclotomic polynomial of order $k$.

# 3 Scott *et al.* method on BLS curves

In this section Scott *et al.* hashing method is applied to BLS curves with embedding degree $k$ equal to $12, 24, 30, 42$ and $48$ respectively. Such an application requires first to determine the cardinality $\tilde{n}(x) \in \mathbb{Q}[x]$ of $\tilde{E}(\mathbb{F}_{p^{k/d(x)}})$, where $d = 6$. Then, one must execute polynomial modular arithmetic as briefly described in the previous section (for further details the reader could refer to Algorithm 2 in [1]).

## 3.1 BLS-12

For BLS curves with $k = 12$, the prime $p$ and the group order $r$ are parameterised by the polynomials:

$$\begin{aligned} p(x) &= \frac{1}{3}(x-1)^2(x^4 - x^2 + 1) + x, \\ r(x) &= x^4 - x^2 + 1. \end{aligned}$$

Since $k/d = 2$, the group $\mathbb{G}_2$ a subgroup of $\tilde{E}(\mathbb{F}_{p^2(x)})$ and the cofactor $c(x)$ is:

$$c(x) = \frac{1}{9}(x^8 - 4x^7 + 5x^6 - 4x^4 + 6x^3 - 4x^2 - 4x + 13). \tag{3.1}$$

For a point $P \in \tilde{E}(\mathbb{F}_{p^2(x)})$, Scott *et al.* method reduces the scalar multiplication $[3c(x)]P$ to

$$[3c(x)]P = [x^3 - x^2 - x + 4]P + [x^3 - x^2 - x + 1]\psi(P) + [-x^2 + 2x - 1]\psi^2(P), \tag{3.2}$$

where $\psi$ is the endomorphism defined in Section 2.1. We consider $[3c(x)]P$ instead of $[c(x)]P$ to ignore the common denominator of 3 that occurs writing $c(x)$ to the base $p(x)$. According to [28, Sec. 8.5], scalar multiplication (3.2) can be computed at the cost of 6 point additions, 2 point doublings, 3 scalar multiplications by the parameter $x$ and 3 applications of $\psi$.

## 3.2 BLS-24

BLS-24 curves are parameterised by the polynomials:

$$p(x) = \frac{1}{3}(x-1)^2(x^8 - x^4 + 1) + x,$$
$$r(x) = x^8 - x^4 + 1.$$

As in the Section 3.1, we consider $[3c(x)]P$ instead of $[c(x)]P$. In this case $\mathbb{G}_2 \subset \tilde{E}(\mathbb{F}_{p^4(x)})$ and the cofactor is a polynomial $c(x)$ of degree 32. Applying Scott *et al.* method, the scalar multiplication $[3c(x)]P$, where $P \in \tilde{E}(\mathbb{F}_{p^4(x)})$, is reduced to

$$[3c(x)]P = [\lambda_0]P + \sum_{i=1}^{6} [\lambda_i]\psi^i(P), \tag{3.3}$$

where $\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6$ are polynomials of $\mathbb{Z}[x]$ with degrees $\leq 8$, and are reported in Appendix A. According to [28, Sec. 8.5], the scalar multiplication (3.3) can be computed at the cost of 21 point additions, 4 point doublings, 8 scalar multiplications by the parameter $x$ and 6 applications of $\psi$.

## 3.3 BLS-30

BLS curves with embedding degree $k = 30$ are parameterised by:

$$p(x) = \frac{1}{3}(x-1)^2(x^{10} - x^5 + 1) + x,$$
$$r(x) = x^8 + x^7 - x^5 - x^4 - x^3 + x + 1.$$

In this case the cofactor is a polynomial $c(x)$ of degree 52 while $\mathbb{G}_2$ is a subgroup of $\tilde{E}(\mathbb{F}_{p^5(x)})$ of order $r(x)$. For a point $P \in \tilde{E}(\mathbb{F}_{p^5(x)})$, one expresses the scalar multiplication $[3c(x)]P$ according to Scott *et al.* as:

$$[3c(x)]P = [\lambda_0]P + \sum_{i=1}^{8} [\lambda_i]\psi^i(P), \tag{3.4}$$

where $\{\lambda_j \mid j = 0, \ldots, 8\}$ are polynomials of $\mathbb{Z}[x]$ of degree $\leq 11$. They are reported in Appendix A. The right hand side of (3.4) can be computed at the cost of 82 point additions, 16 point doublings, 11 scalar multiplications by the parameter $x$ and 67 applications of $\psi$.

## 3.4 BLS-42

In the case of BLS curves with $k = 42$, the group $\mathbb{G}_2 = \tilde{E}(\mathbb{F}_{p^{7(x)}}) \cap \tilde{E}[r(x)]$, where:

$$p(x) = \frac{1}{3}(x-1)^2(x^{14} - x^7 + 1) + x,$$
$$r(x) = x^{12} + x^{11} - x^9 - x^8 + x^6 - x^4 - x^3 + x + 1.$$

The cofactor is parameterised by a polynomial $c(x)$ of degree 100. Writing it to the base $p(x)$, the scalar multiplication $[3c(x)]P$, with $P \in \tilde{E}(\mathbb{F}_{p^{7(x)}})$, is reduced to

$$[3c(x)]P = [\lambda_0]P + \sum_{i=1}^{12} [\lambda_i]\psi^i(P), \tag{3.5}$$

where $\{\lambda_j \mid j = 0, \ldots, 12\}$ are polynomials of $\mathbb{Z}[x]$ with degrees $\leq 15$, and are reported in Appendix A. Then (3.5) can be computed at the cost of 151 point additions, 54 point doublings, 15 scalar multiplications by the parameter $x$ and 125 applications of $\psi$.

## 3.5 BLS-48

For BLS curves with $k = 48$, the prime $p$ and the group order $r$ are parameterised by the polynomials:

$$p(x) = \frac{1}{3}(x-1)^2(x^{16} - x^8 + 1) + x,$$
$$r(x) = x^{16} - x^8 + 1.$$

The cofactor $c(x)$ is a polynomial of degree 128 and $\mathbb{G}_2$ is a subgroup of $\tilde{E}(\mathbb{F}_{p^{8(x)}})$. Given some rational point $P \in \tilde{E}(\mathbb{F}_{p^{8(x)}})$, Scott *et al.* method reduces the scalar multiplication $[3c(x)]P$ to

$$[3c(x)]P = [\lambda_0]P + \sum_{i=1}^{14} [\lambda_i]\psi^i(P), \tag{3.6}$$

where $\{\lambda_j \mid j = 0, \ldots, 14\}$ are polynomials of $\mathbb{Z}[x]$ with degrees $\leq 16$, and are reported in Appendix A. As in previous cases, we consider $[3c(x)]P$ instead of $[c(x)]P$. The scalar multiplication (3.6) can be computed at the cost of 132 point additions, 120 point doublings, 16 scalar multiplications by the parameter $x$ and 130 applications of $\psi$.

# 4 Fuentes *et al.* method on BLS curves with $k = 12, 24, 30$

In this section we apply Fuentes *et al.* hashing method to BLS for $k$ equal to 12, 24 and 30. We have already noticed that this method requires an expensive pre-computation in order to obtain the polynomial $h(z)$. This was infeasible when $k \in \{42, 48\}$ for the computational power at our disposal. For this reason, we do not consider these cases in this section and we will address them with a different approach in Section 5.

## 4.1 BLS-12

For BLS curves with $k = 12$, the value $a$ in (2.5) is parameterised by the following polynomial in $x$:

$$a(x) = \frac{1}{2}\left(t(x) + f(x)\frac{\tilde{t}(x) - 2}{\tilde{f}(x)}\right) \equiv \frac{25}{299}x^{11} - \frac{25}{69}x^{10} + \frac{508}{897}x^9 - \frac{268}{897}x^8 - \frac{112}{897}x^7 +$$

$$+\frac{586}{897}x^6 - \frac{518}{897}x^5 - \frac{126}{299}x^4 + \frac{367}{299}x^3 - \frac{215}{897}x^2 + \frac{64}{299}x + \frac{41}{69} \pmod{\tilde{n}(x)}.$$

Reducing the matrix

$$M = \left[\begin{array}{cc|ccc} & c(x) & 0 & 0 & 0 \\ -a(x) & \bmod c(x) & 1 & 0 & 0 \\ -a^2(x) & \bmod c(x) & 0 & 1 & 0 \\ -a^3(x) & \bmod c(x) & 0 & 0 & 1 \end{array}\right]$$

using generalised LLL [33], we obtain

$$M' = \begin{bmatrix} -x+1 & -2 & x-1 & x^2-x+1 \\ -2 & 0 & x^2-x+1 & x-1 \\ 0 & x^2-x-1 & x-1 & 2 \\ x^2-x-1 & x-1 & 2 & 0 \end{bmatrix}.$$

The polynomial $h(z)$ can be defined from the 4-th row of $M'$ as

$$h(z) = \sum_{i=1}^{4} M'(4,i)z^{i-1} = (x^2 - x - 1) + (x-1)z + 2z^2 \tag{4.1}$$

and so

$$h(a(x)) = (x^2 - x - 1) + (x-1)a(x) + 2a^2(x) \equiv (3x^2 - 3)c(x) \pmod{\tilde{n}(x)},$$

with $\gcd(3x^2 - 3, r(x)) = 1$. Hence, if $P \in \tilde{E}(\mathbb{F}_{p^{2(x)}})$, then $[h(a(x))]P$ is a multiple of $[c(x)]P$. In particular:

$$[h(a(x))]P = h(\psi)P = [x^2 - x - 1]P + [x-1]\psi(P) + [2]\psi^2(P). \tag{4.2}$$

This can be computed at the cost of 5 point additions, 1 point doubling, 2 scalar multiplications by the parameter $x$ and 3 applications of $\psi$.

## 4.2 BLS-24

As in the previous case, for the BLS curves with $k = 24$, we obtain a polynomial $a(x)$ of degree 39, and $h(z)$ defined as:

$$h(z) = (x^4 - x^3 - 1) + (x^3 - x^2)z + (x^2 - x)z^2 + (x-1)z^3 + 2z^4, \tag{4.3}$$

where $h(a(x))$ is congruent to $(3x^4 - 3)c(x)$ modulo $\tilde{n}(x)$. Since $\gcd(3x^4 - 3, r(x)) = 1$, then $[h(a(x))]P$, for $P \in \tilde{E}(\mathbb{F}_{p^{4(x)}})$, is a point of $\mathbb{G}_2$, and

$$[h(a(x))]P = [x^4 - x^3 - 1]P + [x^3 - x^2]\psi(P) + [x^2 - x]\psi^2(P) + [x-1]\psi^3(P) + [2]\psi^4(P). \tag{4.4}$$

To compute the image of $P$, it requires 9 point additions, 1 point doubling, 4 scalar multiplications by $x$ and 10 applications of the endomorphism $\psi$.

## 4.3 BLS-30

In the case of BLS curves with embedding degree $k = 30$, Fuentes *et al.* method constructs a polynomial $a(x)$ of degree equal to 59 and to a polynomial $h(z)$ defined by

follows:

$$\begin{aligned} h(z) = &(x^5 - x^4 - 1) + (-x^5 + 2x^4 - x^3 + 1)z + (x^5 - 2x^4 + 2x^3 - x^2 - 1)z^2 + \\ &+ (x^4 - 2x^3 + 2x^2 - x)z^3 + (x^3 - 2x^2 + 2x - 1)z^4 + (x^2 - 2x + 3)z^5 + \\ &+ (x - 3)z^6 + 2z^7, \end{aligned} \tag{4.5}$$

where $h(a(x))$ is congruent to $(3x^5 - 3)c(x)$ modulo $\tilde{n}(x)$. Hence

$$\begin{aligned} [h(a(x))]P = \quad &[x^5 - x^4 - 1]P + [-x^5 + 2x^4 - x^3 + 1]\psi(P) + \\ &+ [x^5 - 2x^4 + 2x^3 - x^2 - 1]\psi^2(P) + [x^4 - 2x^3 + 2x^2 - x]\psi^3(P) + \\ &+ [x^3 - 2x^2 + 2x - 1]\psi^4(P) + [x^2 - 2x + 3]\psi^5(P) + [x - 3]\psi^6(P) + \\ &+ [2]\psi^7(P) \end{aligned} \tag{4.6}$$

returns a point $P \in \mathbb{G}_2 = \tilde{E}(\mathbb{F}_{p^{5(x)}}) \cap \tilde{E}[r(x)]$, for $P \in \tilde{E}(\mathbb{F}_{p^{5(x)}})$, since $\gcd(3x^5 - 3, r(x)) = 1$. The formula (4.6) can be computed at the cost of 25 point additions, 2 point doubling, 5 scalar multiplications by the parameter $x$ and 27 applications of $\psi$.

# 5 Faster hash maps for BLS curves with $k = 42, 48$

In the previous section, we observed that the degree of the polynomial $a(x)$ grows when $k$ grows. The results in Section 3 show that the same holds for $c(x)$ too. This affects the sizes of the polynomials composing the matrix $M$ and the computational cost to reduce it. The computational power at our disposal did not allow us to complete the application of Fuentes *et al.* method to BLS curves with $k = 42$ and $k = 48$. However, in this section we give two formulas $\mathfrak{h}(z)$ obtained without the use of the LLL algorithm. We begin considering the case $k = 48$.

## 5.1 BLS-48

We note that two of the polynomials $h(z)$ obtained in the previous section, precisely (4.1) and (4.3), satisfy:

(i) $\deg(h(z)) = k/6$,

(ii) $h_{k/6}(x) = 2$,

(iii) $h_i(x) = x^{\deg(h(z))-i} - x^{\deg(h(z))-i-1}$, for $0 < i < k/6$,

(iv) $h_0(x) = x^{\deg(h(z))} - x^{\deg(h(z))-1} - 1$.

For $k = 48$, we define the polynomial

$$\mathfrak{h}(z) = (x^8 - x^7 - 1) + \sum_{i=1}^{7}(x^{8-i} - x^{7-i})z^i + 2z^8.$$

This polynomial satisfies (i),(ii),(iii) and (iv), and conditions (CI), (CII) as proved below.

**Proposition 1.** *For a BLS curve with $k = 48$ defined over $\mathbb{F}_{p(x)}$, the polynomial*

$$\mathfrak{h}(z) = (x^8 - x^7 - 1) + \sum_{i=1}^{7}(x^{8-i} - x^{7-i})z^i + 2z^8,$$

*satisfies the two conditions:*

- *$[\mathfrak{h}](\psi)P$ is a multiple of $[c(x)]P$ for all $P \in \tilde{E}(\mathbb{F}_{p^{k/d(x)}})$;*
- *the coefficients $\mathfrak{h}_i$ of $\mathfrak{h}(z)$ satisfy $\deg(\mathfrak{h}_i(x)) \leq \deg(c(x))/\varphi(k)$ for all $i$.*

*The point*

$$[\mathfrak{h}(\psi)]P = [x^8 - x^7 - 1]P + \sum_{i=1}^{7}[x^{8-i} - x^{7-i}]\psi^i(P) + [2]\psi^8(P) \tag{5.1}$$

*belongs to $\mathbb{G}_2 = \tilde{E}(\mathbb{F}_{p^8(x)}) \cap \tilde{E}[r(x)]$ for every $P \in \tilde{E}(\mathbb{F}_{p^8(x)})$.*

*Proof.* Consider the parametrization of BLS-48 curves from Section 3.5. Let $\tilde{n}(x), f(x)$, $\tilde{f}(x), t(x)$ and $\tilde{t}(x)$ be the polynomial prametrization of $\tilde{n}, f, \tilde{f}, t$ and $\tilde{t}$ respectively. In particular, we verified by direct computation that $\gcd(\tilde{f}(x), \tilde{n}(x)) = 1$. This ensures that (see (2.6))

$$a(x) = \frac{t(x)}{2} - \frac{f(x)(\tilde{t}(x) - 2)}{2\tilde{f}(x)} \mod \tilde{n}(x)$$

is a polynomial with rational coefficients. By direct computation, the following relation holds:

$$\mathfrak{h}(a(x)) \equiv 3(x^8 - 1)c(x) \mod \tilde{n}(x).$$

Let $x_0 \in \mathbb{Z}$ and let $a$ and $c$ be equal to $a(x_0)$ and $c(x_0)$ respectively. Then $\mathfrak{h}(a) \equiv 0 \mod c$. As $\psi P = [a]P$, it follows that $[\mathfrak{h}(\psi)P] = [\mathfrak{h}(a)]P$ is a multiple of $[c]P$, and therefore $[\mathfrak{h}(\psi)]P \in \mathbb{G}_2$. In Appendix B we provide the *magma* [34] code used to verify this proof.

Let $\mathfrak{h}_0(x), \ldots, \mathfrak{h}_8(x)$ denote the coefficients of $\mathfrak{h}(z)$. It is easy to see that $\deg(\mathfrak{h}_i(x)) \leq \deg(c(x))/\varphi(k)$ for all $i \in \{0, \ldots, 8\}$, since $c(x)$ has degree 128 and $\varphi(48) = 16$. □

The right hand side of (5.1) can be computed at the cost of 17 point additions, 1 point doubling, 8 scalar multiplications by the parameter $x$ and 36 applications of $\psi$.

## 5.2 BLS-42

The same approach does not work for BLS curves with $k = 42$. Indeed, for $k = 42$, the polynomial

$$\xi(z) = (x^7 - x^6 - 1) + \sum_{i=1}^{6}(x^{8-i} - x^{7-i})z^i + 2z^7$$

satisfies conditions (i),(ii),(iii) and (iv), but $\xi(a(x))$ is not a multiple of $c(x)$. However, we observed that the following relation holds:

$$\xi(a(x))/c(x) = 3(x^7 - 1)/(x^2 - x + 1).$$

Let $\mathfrak{h}(z) = (x^2 - x + 1)((x^7 - x^6 - 1) + \sum_{i=1}^{6}(x^{8-i} - x^{7-i})z^i + 2z^7)$, we were able to obtain a multiple of $c(x)$ that *almost* satisfies the two conditions (CI), (CII). This is specified in the following proposition.

**Proposition 2.** *For a BLS curve $E$, defined over $\mathbb{F}_{p(x)}$ with $k = 42$, the polynomial*

$$\begin{aligned}\mathfrak{h}(z) = &\left(x^9 - 2x^8 + 2x^7 - x^6 - x^2 + x - 1\right) + \\ &+ \sum_{i=1}^{6}\left(x^{9-i} - 2x^{8-i} + 2x^{7-i} - x^{6-i}\right)z^i + \left(2x^2 - 2x + 2\right)z^7\end{aligned} \tag{5.2}$$

*is such that:*

- $\mathfrak{h}(\psi)P$ *is a multiple of* $[c(x)]P$ *for all* $P \in \tilde{E}(\mathbb{F}_{p^{k/d}(x)})$*;*
- *the coefficients* $\mathfrak{h}_i$ *of* $\mathfrak{h}(z)$ *satisfy* $\deg(\mathfrak{h}_i(x)) \leq \deg(c(x))/\varphi(k)$ *for all* $i \neq 0$*;*
- *the constant term* $\mathfrak{h}_0$ *of* $\mathfrak{h}(z)$ *has degree equal to* $\lfloor \deg(c(x))/\varphi(k) \rfloor + 1$.

*Hence, the point*

$$\begin{aligned}[h(\psi)]P = &\left[x^9 - 2x^8 + 2x^7 - x^6 - x^2 + x - 1\right]P \\ &+ \sum_{i=1}^{6}\left[x^{9-i} - 2x^{8-i} + 2x^{7-i} - x^{6-i}\right]\psi^i(P) + \left[2x^2 - 2x + 2\right]\psi^7(P)\end{aligned} \tag{5.3}$$

*belongs to* $\mathbb{G}_2 = \tilde{E}(\mathbb{F}_{p^{7(x)}}) \cap \tilde{E}[r(x)]$*, for every* $P \in \tilde{E}(\mathbb{F}_{p^{7(x)}})$.

*Proof.* We follow here the same reasoning as in the proof of Proposition 1. Consider the parametrization of Section 3.4. The following relation holds:

$$\mathfrak{h}(a(x)) \equiv 3(x^7 - 1)c(x) \mod \tilde{n}(x).$$

Therefore, $\mathfrak{h}(a(x))$ is a non-zero multiple of $c(x)$, and $[\mathfrak{h}(\psi)]P \in \mathbb{G}_2$. We provide in Appendix B the *magma* code that we used to make the computational verifications of this proof.

Denoting with $\mathfrak{h}_0(x), \ldots, \mathfrak{h}_7(x)$ the coefficients of $\mathfrak{h}(z)$, it could be observed that $\deg(\mathfrak{h}_i(x)) \leq \deg(c(x))/\varphi(k)$ for all $i \in \{1, \ldots, 7\}$, since $c(x)$ has degree 100 and $\varphi(42)$ is equal to 12. The degree of $\mathfrak{h}_0(x)$ is equal to $\lfloor \deg(c(x))/\varphi(k) \rfloor + 1$. □

The right hand side of (5.3) can be computed at the cost of 33 point additions, 1 point doubling, 9 scalar multiplications by the parameter $x$ and 42 applications of $\psi$. Note that $\mathfrak{h}(z)$ does not fully satisfies condition (CII), since $\mathfrak{h}_0(x) = x^9 - 2x^8 + 2x^7 - x^6 - x^2 + x - 1$ is of degree 9, instead of being $\leq \lfloor \deg(c(x))/\varphi(k) \rfloor = 8$.

# 6 Comparisons and conclusions

In Table 6.1, we present an efficiency comparison between the hash maps into $\mathbb{G}_2$ from the previous three sections. The second column refers to the hash maps obtained by applying Scott *et al.* method (see Section 3). The third column contains computational costs of the hash maps obtained by applying *Fuentes et al.* method (see Section 4). The last column reports the costs of the hash maps we proposed in Section 5. With 'A' we denote a point addition, with 'D' a point doubling, with 'Z' a scalar multiplication by the parameter $x$ and with '$\psi$' an application of the endomorphism $\psi$.

| Curve | Scott et al. | Fuentes et al. | Our proposals |
|---|---|---|---|
| BLS-12 | 6A 2D 3Z $3\psi$ | 5A 1D 2Z $3\psi$ | |
| BLS-24 | 21A 4D 8Z $6\psi$ | 9A 1D 4Z $10\psi$ | |
| BLS-30 | 82A 16D 11Z $67\psi$ | 25A 2D 5Z $27\psi$ | |
| BLS-42 | 151A 54D 15Z $125\psi$ | | 33A 1D 9Z $42\psi$ |
| BLS-48 | 132A 120D 16Z $130\psi$ | | 17A 1D 8Z $36\psi$ |

Table 6.1: Comparison between the computational cost of each hash map.

We underline that, in each hashing maps, the multiplication by $x$ dominates the other operations. In fact, the algorithms to compute large scalar multiplications generally require many point additions and doublings. Furthermore, the endomorphism $\psi$ can be efficiently computed.

For $k = 12, 24$ and 30, the hash map found following Fuentes *et al.* method turned out to be more efficient than the ones found with Scott *et al.* method. In particular, for $k = 12$ we see a 3/2-fold improvement, for $k = 24$ the hash map is twice as fast as that of Scott *et al.*, while for $k = 30$ the hash map determines a 11/5-fold improvement.
For BLS curves with $k = 42$, our proposed hash map leads to a 15/9-fold improvement over Scott *et al.* method. Finally, for $k = 48$, our hash map is twice faster than the one obtained according to Scott *et al.*

Using the *Apache Milagro Crypto Library* [35], we implemented the hash maps (3.2) and (4.2) for BLS with $k = 12$ obtained by applying Scott *et al.* and Fuentes *et al.* method respectively. In Table 6.2 we summarise the timing results of two benchmark tests for the two hash maps run on two different devices.

| Processor | Scott et al. | Fuentes et al. |
|---|---|---|
| Intel(R) Core(TM) i5-5257U 64-bit - 2.7 GHz | 2.83 ms | 1.98 ms |
| Quad-core ARM Cortex A53 64-bit - 1.2 GHz | 50.26 ms | 35.88 ms |

Table 6.2: *Each value corresponds to the average time (in milliseconds) considered for each hash from a sample of 1000 hashes.*

These experimental results show that the hashing map obtained with Fuentes *et al.* method is approximately 30% faster than the map obtained with Scott *et al.* method, as we expected from the estimations in Table 6.1.

# Bibliography

[1] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and E. J. Kachisa, "Fast Hashing to G2 on Pairing-Friendly Curves," in *Pairing-Based Cryptography – Pairing 2009*, vol. 5671 of *LNCS*, pp. 102–113, Springer, 2009.

[2] L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez, "Faster Hashing to G2," in *Selected Areas in Cryptography*, vol. 7707 of *LNCS*, pp. 412–430, Springer, 2012.

[3] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing Elliptic Curve Logarithms

to Logarithms in a Finite Field," in *IEEE Transactions on Information Theory*, vol. 39, pp. 1639–1646, IEEE, 1993.

[4] G. Frey and H.-G. Rück, "A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves," in *Mathematics of Computation*, vol. 62, pp. 865–874, American Mathematical Society, 1994.

[5] A. Joux, "A One Round Protocol for Tripartite Diffie–Hellman," in *Algorithmic Number Theory*, vol. 1838 of *LNCS*, pp. 385–393, Springer, 2000.

[6] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems Based on Pairings," in *Proceedings of the 2000 Symposium on Cryptography and Information Security*, SCIRP, 2000.

[7] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," in *Advances in Cryptology — CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 213–229, Springer, 2001.

[8] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[9] P. S. L. M. Barreto, S. Galbraith, C. O. Éigeartaigh, and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties," *Designs, Codes and Cryptography*, vol. 42, no. 3, pp. 239–271, 2007.

[10] F. Hess, N. Smart, and F. Vercauteren, "The Eta Pairing Revisited," in *IEEE Transactions on Information Theory*, vol. 52, pp. 4595–4602, IEEE, 2006.

[11] F. Hess, "Pairing Lattices," in *International Conference on Pairing-Based Cryptography*, vol. 5209 of *LNCS*, pp. 18–38, 2008. Springer.

[12] D. Freeman, M. Scott, and E. Teske, "A Taxonomy of Pairing-Friendly Elliptic Curves," *Journal of Cryptology*, vol. 23, no. 2, pp. 224–280, 2010.

[13] R. Balasubramanian and N. Koblitz, "The Improbability That an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes–Okamoto–Vanstone Algorithm," *Journal of Cryptology*, vol. 11, no. 2, pp. 141–145, 1998.

[14] F. Luca, D. J. Mireles, and I. E. Shparlinski, "MOV Attack in Various Subgroups on Elliptic Curves," *Illinois Journal of Mathematics*, vol. 48, no. 3, pp. 1041–1052, 2004.

[15] A. Miyaji, M. Nakabayashi, and S. Takano, "New Explicit Conditions of Elliptic Curve Traces for FR-reduction," in *IEICE Transactions on Fundamentals of*

*Electronics, Communications and Computer Sciences*, vol. E84–A, pp. 1234–1243, IEICE, 2001.

[16] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing Elliptic Curves with Prescribed Embedding Degrees," in *Security in Communication Networks*, vol. 2576 of *LNCS*, pp. 257–267, Springer, 2003.

[17] F. Brezing and A. Weng, "Elliptic Curves Suitable for Pairing Based Cryptography," *Designs, Codes and Cryptography*, vol. 37 (1), pp. 133–141. 2005.

[18] P. S. L. M. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order," in *Selected Areas in Cryptography*, vol. 3897 of *LNCS*, pp. 319–331, Springer, 2006.

[19] D. Freeman, "Constructing Pairing-Friendly Elliptic Curves with Embedding Degree 10," in *Algorithmic Number Theory*, vol. 4076 of *LNCS*, pp. 452–465, Springer, 2006.

[20] E. J. Kachisa, E. F. Schaefer, and M. Scott, "Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field," in *Pairing-Based Cryptography – Pairing 2008*, vol. 5209 of *LNCS*, pp. 126–135, Springer, 2008.

[21] K. Rubin and A. Silverberg, "Choosing the Correct Elliptic Curve in the CM Method," *Mathematics of Computation*, vol. 79 (269), pp. 545–561. 2010.

[22] E. Brier, J. S. Coron, T. Icart, D. Madore, H. Randriam, and M. Tibouchi, "Efficient Indifferentiable Hashing into Ordinary Elliptic Curves," in *Annual Cryptology Conference*, vol. 6223 of *LNCS*, pp. 237–254, Springer, 2010.

[23] R. Barbulescu, P. Gaudry, and T. Kleinjung, "The Tower Number Field Sieve," in *Advances in Cryptology – ASIACRYPT 2015*, vol. 9453 of *LNCS*, pp. 31–55, Springer, 2015.

[24] T. Kim and R. Barbulescu, "Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case," in *Advances in Cryptology – CRYPTO 2016*, vol. 9814 of *LNCS*, pp. 543–571, Springer, 2016.

[25] T. Kim and J. Jeong, "Extended Tower Number Field Sieve with Application to Finite Fields of Arbitrary Composite Extension Degree," in *IACR International Workshop on Public Key Cryptography*, vol. 10174 of *LNCS*, pp. 388–408, 2017. Springer.

[26] A. Menezes, P. Sarkar, and S. Singh, "Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-based Cryptography," in *Proceedings of Mycrypt*, vol. 10311 of *LNCS*, pp. 83–108, 2016.

[27] R. Barbulescu and S. Duquesne, "Updating Key Size Estimations for Pairings," *Journal of Cryptology*, vol. 32, pp. 1298–1336, 2017.

[28] N. El Mrabet and M. Joye, *Guide to Pairing-Based Cryptography*. Cryptography and Network Security, Chapman & Hall/CRC, 1st ed., 2017.

[29] R. P. Gallant, R. J. Lambert, and S. A. Vanstone, "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms," in *Annual International Cryptology Conference*, vol. 2139 of *LNCS*, pp. 190–200, 2001.

[30] S. D. Galbraith and M. Scott, "Exponentiation in pairing-friendly groups using homomorphisms," in *International Conference on Pairing-Based Cryptography*, vol. 5209 of *LNCS*, pp. 211–224, Springer.

[31] A. K. Lenstra, H. W. Lenstra, and L. Lovasz, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[32] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems.*, vol. 234 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer, 1997.

[33] S. Paulus, "Lattice Basis Reduction in Function Fields," in *International Algorithmic Number Theory Symposium*, vol. 1423 of *LNCS*, pp. 567–575, Springer, 1998.

[34] W. Bosma, J. Cannon, and C. Playoust, "The Magma algebra system. I. The user language," *Journal of Symbolic Computation*, vol. 24, no. 3-4, pp. 235–265, 1997.

[35] Miracl-Apache, "Apache Milagro Crypto Library (AMCL)." https://github.com/apache/incubator-milagro-crypto-c.

# Appendix A

In the following we report the polynomials in $x$ which are the coefficients of the hash maps obtained by applying Scott *et al.* method to BLS curves having $k = 24, 30, 42, 48$.

## BLS-24

Given a rational point $P \in \tilde{E}(\mathbb{F}_{p^4(x)})$, the map (3.3) sends $P$ into the element $[\lambda_0]P + \sum_{i=1}^{6}[\lambda_i]\psi^i(P)$ of $\mathbb{G}_2$, where:

$$\begin{aligned}
\lambda_0 =& -2x^8 + 4x^7 - 3x^5 + 3x^4 - 2x^3 - 2x^2 + x + 4,\\
\lambda_1 =& x^5 - x^4 - 2x^3 + 2x^2 + x - 1,\\
\lambda_2 =& x^5 - x^4 - x + 1,\\
\lambda_3 =& x^5 - x^4 - x + 1,\\
\lambda_4 =& -3x^4 + x^3 + 4x^2 + x - 3,\\
\lambda_5 =& 3x^3 - 3x^2 - 3x + 3,\\
\lambda_6 =& -x^2 + 2x - 1.
\end{aligned}$$

## BLS-30

The map (3.4) sends $P \in \tilde{E}(\mathbb{F}_{p^5(x)})$ into the element $[\lambda_0]P + \sum_{i=1}^{8}[\lambda_i]\psi^i(P) \in \mathbb{G}_2$, with:

$$\begin{aligned}
\lambda_0 &= x^{11} - x^{10} - 2x^9 + 3x^8 + 2x^7 - 3x^6 - x^5 + 2x^4 - x^3 + 4x^2 + x + 7,\\
\lambda_1 &= x^{11} - 3x^{10} + 3x^9 + x^8 - 5x^7 + x^6 + 4x^5 - x^4 - 4x^3 + 4x^2 - 8x - 11,\\
\lambda_2 &= -x^{10} + 4x^9 - 6x^8 + 5x^7 - 2x^6 + 2x^5 - 5x^4 + 4x^3 - 3x + 11,\\
\lambda_3 &= x^8 - 2x^7 + 2x^6 - x^5 - x^4 + 2x^3 - 2x^2 + x,\\
\lambda_4 &= x^8 - 2x^7 + 2x^6 - x^5 - x^3 + 2x^2 - 2x + 1,\\
\lambda_5 &= -4x^7 + 3x^6 + 2x^5 - x^4 - x^3 + 2x^2 + 3x - 4,\\
\lambda_6 &= 6x^6 - 7x^5 - 3x^4 + 8x^3 - 3x^2 - 7x + 6,\\
\lambda_7 &= -4x^5 + 8x^4 - 4x^3 - 4x^2 + 8x - 4,\\
\lambda_8 &= x^4 - 3x^3 + 4x^2 - 3x + 1.
\end{aligned}$$

## BLS-42

The map (3.5) sends $P \in \tilde{E}(\mathbb{F}_{p^{7(x)}})$ into the element $[\lambda_0]P + \sum_{i=1}^{12}[\lambda_i]\psi^i(P) \in \mathbb{G}_2$, with:

$$\begin{aligned}
\lambda_0 &= -4x^{15} + 7x^{14} - x^{13} - 4x^{12} + 4x^{11} + 2x^{10} - 4x^9 + 5x^8 - 4x^7 - 2x^6 + 2x^5 \\
&\quad - 2x^4 - 4x^3 + 9x^2 + 5x + 9, \\
\lambda_1 &= 6x^{15} - 7x^{14} - 9x^{13} + 15x^{12} - 14x^{10} + 7x^9 - 2x^8 - 5x^7 + 13x^6 - 3x^5 \\
&\quad - 7x^4 + 11x^3 + 6x^2 - 22x - 19, \\
\lambda_2 &= -7x^{14} + 15x^{13} - 4x^{12} - 14x^{11} + 15x^{10} + 2x^9 - 13x^8 + 19x^7 - 9x^6 - 14x^5 \\
&\quad + 15x^4 - 16x^2 + 4x + 22, \\
\lambda_3 &= 2x^{13} - 6x^{12} + 6x^{11} + x^{10} - 8x^9 + 8x^8 - 3x^7 - 9x^6 + 12x^5 + 2x^4 - 13x^3 \\
&\quad + 10x^2 + 4x - 6, \\
\lambda_4 &= -x^{12} + 4x^{11} - 6x^{10} + 5x^9 - 2x^8 + 3x^5 - 7x^4 + 5x^3 + x^2 - 5x + 3, \\
\lambda_5 &= x^{10} - 2x^9 + 2x^8 - x^7 - x^4 + 2x^3 - 2x^2 + x, \\
\lambda_6 &= x^{10} - 2x^9 + 2x^8 - x^7 - x^3 + 2x^2 - 2x + 1, \\
\lambda_7 &= -6x^9 - 2x^8 + 2x^7 + 6x^6 + 6x^3 + 2x^2 - 2x - 6, \\
\lambda_8 &= 15x^8 + 5x^7 - 19x^6 - 8x^5 + 14x^4 - 8x^3 - 19x^2 + 5x + 15, \\
\lambda_9 &= -20x^7 + 5x^6 + 30x^5 - 15x^4 - 15x^3 + 30x^2 + 5x - 20, \\
\lambda_{10} &= 15x^6 - 16x^5 - 12x^4 + 26x^3 - 12x^2 - 16x + 15, \\
\lambda_{11} &= -6x^5 + 12x^4 - 6x^3 - 6x^2 + 12x - 6, \\
\lambda_{12} &= x^4 - 3x^3 + 4x^2 - 3x + 1.
\end{aligned}$$

## BLS-48

The map (3.6) sends $P \in \tilde{E}(\mathbb{F}_{p^{8(x)}})$ into the element $[\lambda_0]P + \sum_{i=1}^{14}[\lambda_i]\psi^i(P)$ of $\mathbb{G}_2$, where:

$$\begin{aligned}
\lambda_0 &= -6x^{16} - 2x^{15} + 8x^{14} + 14x^{13} - 14x^{11} - 8x^{10} + 3x^9 + 11x^8 + 8x^7 - 14x^5 \\
&\quad - 14x^4 + 8x^2 + 5x + 4, \\
\lambda_1 &= 10x^{15} + 6x^{14} - 26x^{13} - 22x^{12} + 22x^{11} + 26x^{10} - 5x^9 - 11x^8 - 16x^7 - 24x^6 \\
&\quad + 10x^5 + 46x^4 + 24x^3 - 16x^2 - 19x - 5,
\end{aligned}$$

$$\lambda_2 = -14x^{14} + 4x^{13} + 34x^{12} - 34x^{10} - 3x^9 + 13x^8 + 24x^6 + 26x^5 - 34x^4 - 56x^3 + 29x + 11,$$

$$\lambda_3 = 8x^{13} - 8x^{12} - 16x^{11} + 16x^{10} + 9x^9 - 9x^8 - 22x^5 - 10x^4 + 40x^3 + 24x^2 - 19x - 13,$$

$$\lambda_4 = -4x^{12} + 8x^{11} - 7x^9 + 3x^8 + 12x^4 - 4x^3 - 20x^2 + 3x + 9,$$

$$\lambda_5 = x^9 - x^8 - 4x^3 + 4x^2 + 3x - 3,$$

$$\lambda_6 = x^9 - x^8 - x + 1,$$

$$\lambda_7 = x^9 - x^8 - x + 1,$$

$$\lambda_8 = -7x^8 - 13x^7 - 8x^6 + 14x^5 + 28x^4 + 14x^3 - 8x^2 - 13x - 7,$$

$$\lambda_9 = 21x^7 + 43x^6 + 6x^5 - 70x^4 - 70x^3 + 6x^2 + 43x + 21,$$

$$\lambda_{10} = -35x^6 - 55x^5 + 34x^4 + 112x^3 + 34x^2 - 55x - 35,$$

$$\lambda_{11} = 35x^5 + 29x^4 - 64x^3 - 64x^2 + 29x + 35,$$

$$\lambda_{12} = -21x^4 + x^3 + 40x^2 + x - 21,$$

$$\lambda_{13} = 7x^3 - 7x^2 - 7x + 7,$$

$$\lambda_{14} = -x^2 + 2x - 1.$$

# Appendix B

## Magma Code for the Verification of Proposition 1

```
P<x> := PolynomialRing(Rationals());

k := 48;
p := ((x -1)^2)*(x^16 -x^8 +1)/3 +x;
r := x^16 -x^8 +1;
t := x +1;
f := Sqrt(-(t^2-4*p)/3);
q := p^8;

// count points on twist over extension
tau := [ P!0: i in [1..9]];
tau[1] := 2;
tau[2] := t;
for i in [2..8] do
   tau[i+1]:= t*tau[i]-p*tau[i-1];
end for;

t_hat := tau[9];
f_hat := Sqrt((-1/3)*(t_hat^2-4*q));
t_tilde := (3*f_hat+t_hat)/2;
f_tilde := Sqrt((-1/3)*(t_tilde^2 - 4*q));
n_tilde := q + 1 -(3*f_hat + t_hat)/2;

// check that GCD(n_tilde, f_tilde) = 1
GCD(n_tilde, f_tilde) eq 1;

c := n_tilde/r;

a := ((1/2)*(t - f*(t_tilde -2)*InverseMod(f_tilde,n_tilde))) mod n_tilde;
//a := ((1/2)*(t + f*(t_tilde -2)*InverseMod(f_tilde,n_tilde))) mod n_tilde;

// if the following is false, one must take the other option for a
(a mod r) eq (p mod r);

c := P!c;
```

```
// write the map
H := [ P!0: i in [1..((k div 6)+1)]];
H[1] := x^(k div 6)-x^((k div 6)-1)-1;
for i in [2..(k div 6)] do
   H[i] := x^((k div 6)+1 -i) - x^((k div 6) -i);
end for;
H[((k div 6)+1)] := 2;

// compute h(a)
h_a :=(H[1] + H[2]*a + H[3]*a^2 + H[4]*a^3 + H[5]*a^4 + H[6]*a^5 + H[7]*a^6 +
    H[8]*a^7 + H[9]*a^8) mod n_tilde;

s := h_a/c;
GCD(P!s,r) eq 1;

// check that h(a) = 0 mod c
(h_a mod c) eq 0;
```

## Magma Code for the Verification of Proposition 2

```
P<x> := PolynomialRing(Rationals());

k := 42;
p := ((x -1)^2)*(x^14 -x^7 +1)/3 +x;
r := P!CyclotomicPolynomial(k);
t := x +1;
f := Sqrt(-(t^2-4*p)/3);
q := p^7;

// count points on twist over extension
tau := [ P!0: i in [1..8]];
tau[1] := 2;
tau[2] := t;
for i in [2..7] do
   tau[i+1]:= t*tau[i]-p*tau[i-1];
end for;

t_hat := tau[8];
f_hat := Sqrt((-1/3)*(t_hat^2-4*q));
t_tilde := (3*f_hat+t_hat)/2;
```

```
f_tilde := Sqrt((-1/3)*(t_tilde^2 - 4*q));
n_tilde := q + 1 -(3*f_hat + t_hat)/2;

// check that GCD(n_tilde, f_tilde) = 1
GCD(n_tilde, f_tilde) eq 1;

c := n_tilde/r;

c := P!c;

a := ((1/2)*(t + f*(t_tilde -2)*InverseMod(f_tilde,n_tilde))) mod n_tilde;
//a := ((1/2)*(t - f*(t_tilde -2)*InverseMod(f_tilde,n_tilde))) mod n_tilde;

// if the following is false, one must take the other option for a
(a mod r) eq (p mod r);

// write the map
C := [ P!0: i in [1..((k div 6)+1)]];
C[1] := (x^(k div 6)-x^((k div 6)-1)-1)*(x^2 - x + 1);
for i in [2..(k div 6)] do
   C[i] := (x^((k div 6)+1 -i) - x^((k div 6) -i))*(x^2 - x + 1);
end for;
C[((k div 6)+1)] := 2*(x^2 - x + 1);

// compute h(a)
h_a :=(C[1] + C[2]*a + C[3]*a^2 + C[4]*a^3 + C[5]*a^4 + C[6]*a^5 + C[7]*a^6 +
    C[8]*a^7) mod n_tilde;

s := h_a/c;
GCD(P!s,r) eq 1;

// check that h(a) = 0 mod c
(h_a mod c) eq 0;
```

# Errata for
# Notes on Lattice-Based Cryptography

**Alessandro Budroni**


Thesis for the degree philosophiae doctor (PhD)
at the University of Bergen

29/06/22 _____ _____

(date and sign. of candidate) (date and sign. of faculty)

# Errata

- page 19, 5 lines from the end, fix line that starts with a dot
- page 19, 4 lines from the end: add blank space missing before 'invertible'
- page 19, 2 lines from the end: fix spacing before \mod
- page 31, line 3: fix paring -> pairing
- page 34, 3 lines from the end: add a reference for 'Scott et al'
- page 50, last line: fix *with* probability
- page 55, line 12: fix equals -> equal
- page 61, 4 lines from the end: remove 'it'
- page 62, 4 lines from the end: discovered -> discover
- page 70, 10 lines from the end: a reference seems to be missing for 'Cheon et al'
- page 70, 5 lines from the end: the Gentry's -> Gentry's
- page 80, 7 lines from the end: the first 'nor' should be a 'neither'
- page 105, just below (5. 2): remove the indentation
- page 108, last line before section 6: 'the the'
- page 109, 8 lines from the end: 'the the'
- page 177, first line: fix triplets -> triples
- page 132, line 3: add missing blank space before 'Let'
- page 137, line 10: Lovasz -> Lov\'asz
- page 141, line 10: add missing blank space before 'In'

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS


uib.no

ISBN: 9788230866252 (print)
9788230860243 (PDF)