UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Lattice-based Cryptography:
Expanding the Design Space

Arnaud Sipasseuth

This thesis is presented as part of the requirements for the conferral of the degree:

Doctor of Philosophy in Computer Science

Supervisor:
Pr. W. Susilo

Co-supervisors:
Dr. T. Plantard & Dr Y. Guomin

The University of Wollongong
School of School of Computing & Information Technology

October, 2020

This research has been conducted with the support of an Australian Government Research Training Program Scholarship.

Declaration

I, *Arnaud Sipasseuth*, declare that this thesis is submitted in partial fulfilment of the requirements for the conferral of the degree *Doctor of Philosophy in Computer Science*, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Arnaud Sipasseuth

March 22, 2021

Abstract

This thesis is a compilation of the main published works I did during my studies in Australia. My research area was lattice-based cryptography, which focuses mainly on a family of mathematical primitives that are supposed to be “quantum-resistant”. The direction of my research was mostly targeted towards constructions that lie outside of the mainly researched lattice forms to provide an alternative direction in the case common constructions were discovered to be insecure. We do have, however, some work that makes use of common constructions in which we expand the design space for better efficiency or security.

At PKC 2008, Plantard et al. published a theoretical framework for a lattice-based signature scheme, namely Plantard-Susilo-Win (**PSW**). Recently, after ten years, we proposed a new signature scheme dubbed the Diagonal Reduction Signature (**DRS**) scheme was presented in the National Institute of Standards and Technology (NIST) PQC Standardization as a concrete instantiation of the initial work. Unfortunately, the initial submission was challenged by Yu and Ducas using the structure that is present on the secret key noise. Thus, we also present a new method to generate random noise in the Diagonal Reduction Signature (**DRS**) scheme to eliminate the aforementioned attack, and all subsequent potential variants. This involves sampling vectors from the n-dimensional ball with uniform distribution. We also give insight on some underlying properties which affects both security and efficiency on the Plantard-Susilo-Win (**PSW**) type schemes and beyond, and hopefully increase the understanding on this family of lattices. This work was published in [SPS20].

In another work, we present a technique to enhance the security of Goldreich, Goldwasser and Halevi (GGH) scheme. The security of GGH has practically been broken by lattice reduction techniques. Those attacks are successful due to the structure of the basis used in the secret key. In this work, we aim to present a new technique to alleviate this problem by modifying the public key which hides the structure of the corresponding private key. We intersect the initial lattice with a random one while keeping the initial lattice as our secret key and use the corresponding result of the intersection as the public key. We show sufficient evidence

that this technique will make GGH implementations secure against the aforementioned attacks. This work was published in [SPS19a].

We also present a novel computational technique to check whether a matrix-vector product is correct with a relatively high probability. While the idea could be related to verifiable delegated computations, most of the literature in this line of work focuses on provably secure functional aspects and do not provide clear computational techniques to verify whether a product $xA = y$ is correct where x, A and y are not given nor computed by the party which requires validity checking: this is typically the case for some cryptographic lattice-based signature schemes. This paper focuses on the computational aspects and the improvement on both speed and memory when implementing such a verifier, and use a practical example: the Diagonal Reduction Signature (DRS) scheme as it was one of the candidates in the recent National Institute of Standards and Technology Post-Quantum Cryptography Standardization Calls for Proposals competition. We show that in the case of DRS, we can gain a factor of 20 in verification speed. This work was published in [SPS19c].

`NewHope` Key Encapsulation Mechanism (KEM) has been presented at USENIX 2016 by Alkim et al. and is one of the remaining lattice-based candidates to the post-quantum standardization initiated by the NIST. However, despite the relative simplicity of the protocol, the bound on the decapsulation failure probability resulting from the original analysis is not tight. We refine this analysis to get a tight upper-bound on this probability which happens to be much lower than what was originally evaluated. As a consequence we propose a set of alternative parameters, increasing the security and the compactness of the scheme. However using a smaller modulus prevent the use of a full NTT algorithm to perform multiplications of elements in dimension 512 or 1024. Nonetheless, similarly to previous works, we combine different multiplication algorithms and show that our new parameters are competitive on a constant time vectorized implementation. Our most compact parameters bring a speed-up of 17% (resp. 11%) in performance but allow to gain more than 19% over the bandwidth requirements and to increase the security of 10% (resp. 7%) in dimension 512 (resp. 1024). This work is available on [PSSZ19] and is still in review.

Each of those works also lead to potential further research, leaving open new challenges. Those will be listed in the last chapter.

Acknowledgments

This thesis is the result of the time spent in Australia. In that regard, there are people I would like to thank and without them this thesis could have gone very awry. Those acknowledgements are multilingual: I believe this should reflect the international melting pot the University of Wollongong currently offers.

First and foremost, I would like to thanks Thomas and Willy for their supervision and patience. It must not have been easy to supervise a student who have a tendency to sidetrack his own research to look at other people's research on top of doing over 400 hours of teaching work. Willy bought us a fridge, Thomas gave us a sofa and it cannot be understated how much it contributed to "6.214"'s productivity. Also special thanks to Yinhao who gave us a *very* comfy chair, and Jongkil who gave me a second screen: this thesis was written faster thanks to you.

Je voudrais aussi remercier Ludovic Perret pour le stage de M1 qui m'a fait découvrir la recherche, Florent Bernard pour m'encourager à poursuivre dedans quand je réfléchissais à me contenter d'un job alimentaire, et Jean-Claude Bajard pour avoir donné à son ancien étudiant l'opportunité d'en faire à plus de 20 heures de vol de lui (au soleil donc).

I would like to thank all the help I have gotten from the UOW societies which helped me a lot when I first came to Australia: ISP, IFIS, UOW Wellbeing, UOW TSA, UOW PPIA, UOW MAS, UOW Matsuri, UOWFS are the first that come to mind but also FOCUS, Fun-Food-Fridays and everything else. Those societies helped me survive in Australia and I am thankful by how much importance UOW gave to mental health and cultural difference.

Saya ingin berterima kasih kepada komunitas Indonesia yang sangat ramah. Saya juga ingin berterima kasih kepada orang-orang Indonesia yang tinggal di dekat 7-eleven. Terima kasih karena sudah memberi saya Indomie, nasi goreng, dan mengajarkan saya tentang kopi luwak.[a] Hidup itu seperti perjalanan di kereta api, dan

[a] Terima kasih karena telah menjual sofa seharga 80 dollar kepada saya, sofa yang kalian dapat

semoga jalur kereta api ini mempunyai pemberhentian di stasiun Jakarta.

Cảm ơn tất cả các bạn việt nam đã không ngần ngại chia sẻ nền văn hoá của mình với mọi người, từ việc đá cầu với bạn cùng nhà của tôi, hay việc chia sẻ những món ăn hấp dẫn như phở chẳng hạn. Tôi rốt cuộc sẽ đến Việt Nam, nhưng ko phải vào mùa hè![b]

感謝臥龍崗大學臺灣學生會歡迎我的加入，並且給我機會練習繁體中文。我也感謝在謝先涵 (Molly) 家的每一場聚會及每一道美食。雖然我的中文仍然不夠好，但我至少從臺灣朋友們身上學到 ” Taiwan No. 1!” [c]

この前はお好み焼きをいただきありがとうございました。本当は広島風が好きなんだけどね笑もう少ししたら日本で働き始めます。アツシが言ってたけど、日本で働くのは大変らしいね。なんで君たちが日本から逃げ出したくなるのか、もうすぐ自分の目で確かめられるよ。なんにせよ、そっちでの生活が楽しみだよ(^^)/[d]

I am not forgetting the Nigerians, the Malaysians, the Belgians, the Dutch, the Brazilians, the British, the Koreans, the Chinese, the Germans, the Indians, the Sri Lankans, the Pakistanis, the Bengalis, the Canadians, the Iranians and whatever nationality of people I have met and not listed. I do not wish to make my acknowledgements longer than a research paper, but I do (currently) have memories with each of you.

I would also like to thanks all the housemates I had, the people who hosted me when I was homeless (in particular Klemens), the "coffee break" mates and the "PhD depression group" of the SMART building. May all of you achieve what you work so hard for.

I also want to thanks all officemates: Jongkil, Yinhao, Clémentine, Tran, Zhunzhun, Cédric, Kaartik and the rest. Also Nicolas who was there but in another building. Vincent, Andréa, et Arthur ont beaucoup égayé la fin de ma thèse et m'ont très certainement motivé à passer bien plus de temps au bureau, sans quoi il serait tout simplement désert.

Vincent, tu m'auras beaucoup appris sur l'histoire de la France, et pas n'importe

dengan gratis.

[b] Happiness is a bowl of Phở

[c] Thank you 遵友, but geology is not a science

[d] Thanks Ren for sharing the "fun" side of Japanese culture!

laquelle, la France du général de Gaulle.[e] Merci de ne pas avoir crié au scandale à chaque bug de mon code que tu as corrigé. Commenter l'actualité politique avec toi est toujours un plaisir: la présidence américaine en particulier nous aura bien fait sourire.

Andréa, tu m'auras illuminé par ta connaissance des mathématiques fondamentales et leur histoire. Sans ton café, mon boulot n'avancerait pas: par transitivité, ma recherche est donc un sous-ensemble de la tienne.

Arthur, tu m'auras fait découvrir les platistes[f], et à quel point je n'aime pas les probas et les stats. Merci surtout de partager avec moi les "extra large snack packs": comme dit le proverbe, le gras qui nous tue pas nous rend plus fort.

Je remercie également Hélène pour les saucissions qu'elle partage et ses histoires sur les fans de l'OM, ce qui me permet de ne pas (trop) regretter la France.

Merci aussi aux amis en France de n'avoir pas oublié que j'existais. Lors des grands moments de solitude, le décalage horaire était plus facile à supporter grâce à vous. En particulier, j'espère que le Hangout durera encore bien longtemps (faute du FCAM)[g].

Merci aussi à ma famille d'avoir installé Signal et de continuer à communiquer via des groupes de discussions. 我感謝我的父母，讓我選擇自己的志向，即使這是一條薪水不豐厚，也不保證未來能找到穩定工作的道路。特別感謝他們在我遠走他鄉的這段時間，仍然對我充滿耐心，並且尊重我的選擇。 À Alexandre, en *espérant* de toi plus d'efforts à l'école et surtout de curiosité: ce n'est pas un vilain défaut!

[e] Le Général de Gaulle n'a-t-il pas dit que toute la France avait été résistante ?

[f] On retiendra que le moyen-âge n'existe pas. C'est une conspiration.

[g] Il semblerait que le FCAM est mort et le Hangout ira le rejoindre d'ici peu. Bon ben... Signal!

Contents

Abstract **iv**

1 Introduction **1**

1.1 Cryptography and the context of this thesis 1

1.1.1 Early days of cryptography 1

1.1.2 Cryptography as an essential war tool 1

1.1.3 "Crypto Wars" and Modern Cryptography 2

1.1.4 Post-Quantum Cryptography 3

1.1.5 This Thesis 3

1.2 Thesis plan 4

2 Background **6**

2.1 Preliminaries 6

2.2 Knapsack 7

2.3 Lattice Basics 8

2.3.1 Lattice, volume and dual 8

2.3.2 Hermite Normal Form and sublattices 10

2.3.3 Norms, lattice constants and heuristics 12

2.4 Computationally hard lattice problems 15

2.4.1 Shortest Vector Problem and variants 15

2.4.2 Closest Vector Problem and variants 18

2.4.3 Shortest Basis Problem and variants 19

2.5 Algorithms to solve lattice problems 21

2.5.1 Gram-Schmidt Orthogonalization algorithm 21

2.5.2 Babai's algorithms 22

2.5.3 Size-Reduction 23

2.5.4 Gauss's algorithm, the case of dimension 2 23

2.5.5 Lenstra-Lenstra-Lovász basis reduction algorithm 24

2.5.6 Enumeration techniques 25

2.5.7 Hermite-Korkine-Zolotarev 28

2.5.8 Block Korkine-Zolotarev Reduction algorithm 28

2.5.9 Kannan's extension technique 29
2.6 Applications to public-key cryptography 30
2.6.1 Encryption and Key Encapsulation Mechanism 31
2.6.2 Public-Key Digital Signature Schemes 32
2.6.3 Knapsack cryptosystems 33
2.6.4 Specific lattices in crypto 35
2.6.5 Other structures and our research direction 38

3 Hiding Lattices within Sublattices 40
3.1 A **HNF** vision of lattice intersections 41
3.2 A **HNF** vision of column permutations 49
3.3 Structure of lattice intersections 52
3.4 Example application: reinforcing **GGH** 54
3.4.1 Modification of **GGH** using an intersecting lattice 54
3.4.2 Modified attack on the intersected **GGH** key 55
3.4.3 Countermeasure by controlling $\omega(S_k)$ and $\omega(R)$ 58
3.4.4 Key sizes comparisons and perfect **HNF** 61
3.5 Security of the reinforced **GGH** 63
3.5.1 Smoothness of determinant 64
3.5.2 Perfectness of basis and primality between factors 65
3.5.3 Shortest Vector and Basis Structure 66
3.5.4 Message Recovery Attacks 69
3.6 Open questions and related work 70
3.6.1 On Ideal Lattices 70
3.6.2 Recovering a vanishing structure 71
3.6.3 Group representation of Gama-Izabachene-Nguyen-Xie 71

4 Diagonal-Dominant Lattice-Based Signatures 73
4.1 The theoretical framework of Plantard-Susilo-Win 74
4.1.1 Before **PSW**: lattices for number representation 74
4.1.2 Spectral Radius and Eigenvalues 75
4.1.3 The original PSW framework 76
4.2 The original **DRS** scheme 79
4.2.1 Setup 80
4.2.2 Signature 83
4.2.3 Verification 85
4.3 On the Security of the Public Key 87
4.3.1 Li, Liu, Nitaj and Pan's Attack 88
4.3.2 Yu and Ducas's Attack 88
4.4 New Setup 90

4.4.1 Picking the Random vectors 92
4.4.2 A Slightly More General Termination Proof 93
4.4.3 On Exploiting the Reduction Capacity for Further Security . . 94
4.4.4 Ensuring the termination of **PSW** 96
4.4.5 Setup Performance 98
4.5 Security estimates 99
4.5.1 BDD-Based Attack 99
4.5.2 Expected Heuristic Security Strength 100
4.5.3 A note on the structure of diagonally dominant lattices 101
4.5.4 A small density comparison with ideal lattices 104

5 Freivalds-accelerated Signature Verification 106
5.1 Freivalds' algorithm 107
5.2 Modifiying Freivalds' technique for lattice-based signature verification 109
5.2.1 The first core idea: Modification of Freivalds' algorithm 109
5.2.2 The second core idea: Changing the verifier 110
5.3 Security considerations 114
5.3.1 Attack models 115
5.3.2 How to choose the primes 117
5.4 Implementation results 120
5.4.1 Time results on a basic implementation 120
5.4.2 Memory storage 121
5.5 Beyond **DRS** and the usage of primes 122
5.5.1 Any random integer ring is fine 122
5.5.2 Other types of basis and lattice-based signature schemes . . . 123

6 Tighter Bounds: Accelerating NewHope 125
6.1 Introduction 126
6.2 Related Art 127
6.3 Notations for this chapter 128
6.3.1 `NewHope` 128
6.3.2 Number Theoretic Transform (NTT) 131
6.4 Analysis of the failure probability 132
6.4.1 About key-encoding and multiplication 132
6.4.2 Distribution of the key-encoding error 133
6.4.3 Estimation of the probability failure 133
6.5 Multiplication strategies for smaller moduli 136
6.5.1 Choose the moduli 136
6.5.2 Mutiplication strategies 137
6.5.3 Mixing the strategies 138

6.6 Global parameters and experimental results 139
6.6.1 Security parameters 139
6.6.2 The particular case of $q = 1601$ 140
6.6.3 Experimental results 140

7 Conclusions and open questions 144
7.1 Using lattice intersections to hide a lattice generated by a secret key . 144
7.2 Using a diagonal dominant matrix to reduce the maximum norm of vectors 144
7.3 Using Freivalds' algorithm to accelerate signature verifications 145
7.4 A potential improvement on New Hope 145

Bibliography 147

A Code snippets 165

Chapter 1

Introduction

1.1 Cryptography and the context of this thesis

1.1.1 Early days of cryptography

In the history of humanity, cryptography has always been valued in conflicts. While steganography, which consists of hiding secret messages in places where the general population does not expect them, cryptography consists of hiding secret messages by transforming them into something which look random in front of unauthorized eyes. One of the oldest form of encryption was to swap each letter by another symbol, namely the “Caesar cipher”, which Julius Caesar used to protect important military messages. All variants of the Caesar cipher were supposedly broken by “frequency analysis” in the 9th century by Al-Kindi. As mentioned in the introduction of Yuanmi Chen’s PhD thesis [Che13], this idea was later used in a work of fiction, namely “The adventure of the dancing men” from Arthur Conan Doyle where Sherlock Holmes successfully applied “frequency analysis” to substitute “dancing men symbols” to letters of the english alphabet. A more complex cipher, the “Vigenére cipher”, based on the original Caesar cipher was conceived by Giovan Battista Bellaso in the 16th century and stayed officially unbroken for at least 3 centuries until Friedrich Kasiski published a general method to break the cipher.

1.1.2 Cryptography as an essential war tool

The real importance of cryptography was highlighted during World War 2 where military tactics, logistics and communication methods reached a point where battles could be arguably won or lost by every slight bit of information (or misinformation). A well-known story, which inspired the historical background behind the movie “The Imitation Game” from 2014, highlighted how British intelligence deployed various assets to attempt a break of the Enigma code used by the Nazis. While Alan Turing was a well-known member of the task force created by the British government to tackle this issue, chess champions and crosswords experts were also part of that effort: the importance of pure mathematics was not highlighted at that time as there was no consensus on how to break a secret code. Nevertheless, the break of the Enigma machine is often referred as a turning point in the war, and even by some as essential to the conclusion of the war.

1.1.3 “Crypto Wars” and Modern Cryptography

In 2019, during the 24th Australasian Conference on Information Security and Privacy (ACISP), Jennifer Seberry gave a retrospective on the history of cryptography during her 50 years in the area [Seb19]. She saw the advent of internet and online trading, and thus the natural and inevitable rise of mathematical methods to conceive cryptosystems and their cryptanalysis (i.e the evaluation of their security with practical and theoretical attacks). The advent of modern cryptography was initiated by Claude Shannon, who promoted the use of mathematics for the formalism of cryptography in a paper he wrote in 1975, effectively transforming cryptography into a science of its own. Back when Jennifer started, cryptography was not widespread. During that period, the US government was investing millions of dollars to keep the ability to break all existing codes, and wanted to make sure foreign agencies would not be able to hide anything from them. The US government was also selling security measures and tools in other countries, but made sure the level of security sold was not secure enough to protect from their eyes, while obviously keeping the higher level of security from themselves. When presenting algorithms and protocols, american scientists were not even allowed to discuss anything beyond the simple statement of the algorithms. Furthermore, when Phil Zimmermann released Pretty Good Privacy (PGP) in 1991 to the world, he was immediately investigated by the US department of justice. While this look like a thing of the past, those so called “Crypto Wars” are still going on, and the message of professor Seberry was, if my memory serves me well, a pledge for public research in cryptography. The well-known leaks provided by Edward Snowden reminded us how the National Security Agency (NSA) deliberately made the NIST promote the use of cryptography they can break[BLN16].

With the advent of fast telecommunications, whether for trading or personal use, cryptography is becoming more and more widespread as malicious attackers, government-based or not, flourish. As Jennifer Seberry reminded us, “without security there is no trust, and without trust there is no business”. The need for privacy has also become increasingly important. To quote Edward Snowden, “Privacy is the right to a free mind. Without privacy, you can't have anything for yourself. Saying you don't care about privacy because you have nothing to hide is like saying you don't care about free speech because you have nothing to say." In 2015, both US president Barack Obama and UK prime minister David Cameron wanted to outlaw backdoor-less cryptography. Thankfully, this did not lead to any change in the legislation.

However, as described by professors Tanja Lange and Daniel Bernstein during a talk at Macquarie University in 2018, a new "nightmare" is coming: quantum computers [BL18].

1.1.4 Post-Quantum Cryptography

The mathematical primitives we use today in cryptography, relying on elliptic curves and number factorization, are mostly broken by quantum computers using Shor's algorithm [Sho97]. On top of that, several theoretical algorithms using quantum algorithms bring significant improvements in the conception of cryptographic attacks as Grover's algorithm for exhaustive search [Gro96]. Both algorithms do require a powerful enough quantum computer which currently does not exist, even though many claim those are coming "soon", especially considering the achievements from the last decade. Nevertheless, research on post-quantum cryptography, i.e the art of conceiving quantum-safe encryption and its analysis, developed as a result of this and is flourishing. In a certain way, the research on post-quantum cryptography is currently peaking with the involvement of the US government.

The NIST has launched a standardization process in order to prepare the world for the next generation of quantum-safe communications, and has encouraged all research institutes to cooperate and focus on this sole objective. Several families of mathematical quantum-safe primitives have been proposed, and three currently stands out: error-correcting codes, multivariate polynomials systems, and integer lattices. Those three families are based on a category of problems that we deem safe for now, i.e the NP-hard problems. While it is not sure if those are really quantum safe, we know they are highly likely to be safer than using co-NP problems (like the integer factorization) as QP=co-NP. Furthermore, if P=NP under quantum reductions, then my next job will probably be inside a fast-food restaurant.

1.1.5 This Thesis

While the work mentioned in this thesis has some involvement with the NIST standardization process for post-quantum cryptography, the work in this thesis is not limited with the involvement of the NIST on academic research. Regardless of the NIST's initiative, public research on post-quantum cryptography still exists and this thesis is part of it. More specifically, this thesis focuses on lattice-based cryptography. While the work described here does not make use on the most popular tools of lattice-based crypto, it is nevertheless a humble contribution.

One part of the work is linked to our recent work on our NIST submission: [PSDS18, SPS19b, SPS20]. Those publications are a description of the work we submitted and a deeper study of the underlying structure following an attack of the initial scheme by Yu and Ducas [YD18a].

Another part of our work is related to the study and usage of unused mathematical primitives to expand the available options for the conception of lattice-based cryptosystems: [SPS19a, SPS19c]. One of them is a framework to enhance the heuristic security of some lattice-based encryption schemes, and the other work is a framework to conditionally enhance the efficiency of some lattice-based signatures.

The last work we present is based on another NIST candidate (NewHope [ERJ+18]): which is still in review after our initial submission to TCHES, then to IEEE TC on 2019-10-05 at IEEE TC and more recently to TETC [PSSZ19]. It is mostly an expansion of the possibilities of NewHope and a test of their efficiency.

Other submissions do exist ([DFK+20, DRS+20] and others offline) but will not appear on this thesis (out of scope).

1.2 Thesis plan

This thesis will be organized in 5 parts. The first part will provide some necessary background on the work presented in this thesis. The next three parts will present some research results, and the last part is a small abstract of the research done and indicates some future research directions.

As someone who studied alongside software engineers, I am fully aware that most of the knowledge online, and by extension most if not all "entry-point" surveys into the state-of-the-art cryptography does not help at all the average implementer into understanding the mathematical properties that are involved. This thesis is written with the hopes that the average reader with a background on basic mathematics such as linear algebra (vector spaces, matrices, rank...) and algebraic structures (groups, rings, polynomials...) should be able to navigate through this thesis just fine. Hopefully, this should avoid many potential footguns.

Chapter 2

Background

2.1 Preliminaries

Before we go deeper into cryptography, we need to give a reminder of the mathematical structures we use and the underlying "hard" problems related to them. Indeed, modern cryptography relies on computational problems to ensure the safety of the secret it aims to protect. We also present in this chapter some of the known techniques to solve those problems.

While this thesis focuses on lattice-based cryptography, I believe mentioning knapsack problems first might be a good idea to remind the historical links between knapsacks cryptosystems and lattices. First as we believe the problems are actually very similar in nature and second, to deal with the apparent inequality of reputation between those two families. Knapsack cryptosystems currently have a bad reputation, enough to be mentioned in the crypto-related card game "Cards Against Cryptography" where one joke card was literally "Knapsack cryptosystems, revisited" [Ano19]. Lattice-based cryptosystems, however, enjoy a remarkable reputation: 28 out of the 82[a] initial submissions for the NIST standardization process were lattice-based. This directly contrasts with the recommendation that Vadim Lyubashevsky gave during his talk in PKC 2016 [Lyu16]:

> *To build practical schemes, it is not enough to just work on "provably-secure" constructions - one needs to understand the underlying knapsack problems*

Granted, our research findings do not base themselves over provably-secure assumptions. Nevertheless, the hardness of lattice problems lie in the construction of hard instances of knapsack, which is where our overall research is building forwards.

[a] count reported in slide 31 of [BL18]

2.2 Knapsack

While the *original knapsack problem* (which is sometimes called the *"0/1 Knapsack Problem"*) was basically how to carry the most valuable items with a limited storage capacity, several versions of the knapsack problem exists.

Definition 1 (Knapsack Problem(s)).
*Let us consider a finite set of tuples $S = \{(w_i, v_i)_{i \in [1,n]}\}$ (i.e (weights,values)) where $S \subset \mathbb{N}^2$ and $W_{max} \in \mathbb{N}$. The three following versions are called Knapsack Problem (**KP**).*
Maximize the value $\sum_{i=1}^{n} x_i v_i$ such that $\sum_{i=1}^{n} x_i w_i \leq W_{max}$ and $\forall i \in [1, n]$:

0/1 KP	$x_i \in \{0, 1\}$
B-Bounded KP	$x_i \in [0, B]$, $B \in \mathbb{N}^*$
Unbounded KP	$x_i \in \mathbb{N}$

Usually when **KP** is mentioned, it is usually the 0/1 version. There is nothing much we will say here about the general knapsack problem as it is a very simple one, aside from it being NP-complete [GJ02]. The knapsack problem we are defining here is the *modular knapsack problem*, which is actually closer to the knapsack problem as defined by Karp [Kar72], and subsequently the *subset-sum problem*. Relaxing the problem by adding a value $M > W_{max}$ and requiring $\sum_{x \in s} x_i w_i = W_{max} \mod M$ essentially gives the modular knapsack problem. The following definition of the modular knapsack problem can be found in [PSZ12] which links the knapsack problem to the subset sum of [LO85].

Definition 2 (Subset-Sum problem).
Given a set of positive integers S and an integer W, is there any non-empty subset $s \subset S$ such that $\sum_{x \in s} x = W$?

Definition 3 (Modular Knapsack problem).
Let $\{X_0, ..., X_d\}$ be a set of positive integers. Let $c = \sum_1^d s_i X_i \mod X_0$, where $s_i \in \{0, 1\}$. A modular knapsack problem is given $\{X_i\}$ and c, find each s_i.

Modular knapsacks are also NP-complete, even by adding multiple instances such that n knapsacks of size $2n$ share a common solution [Nie90]. Note that the knapsack problem doesn't have to rely exclusively on integers: all it needs is a group (and its associated operator) and multiplicative knapsacks do exist [CR88].

Another parameter for knapsacks instances is the density:

Definition 4 (Knapsack density).
We say a knapsack on a set of integers S has density $d = |S| / \max(\{\log_2(x) \mid x \in S\})$.

This is the most basic definition of the knapsack density, which can be found in [LO85]. The literature sometimes uses different definitions such as in [NS05] and there has been an attempt to unify the different density definitions with the help of a Hamming weight [Kun08]. When the density $d > 1$ there is in general multiple solutions to the same instance. This definition is mostly relevant in the field of cryptography for $d \leq 1$ to encode an unique solution and we will leave this as it is for now. We will mention a bit the history of knapsack cryptosystems and how lattices are involved in the next section.

2.3 Lattice Basics

Lattices are a popular object used for non-cryptographic problems especially for the representation of natural physics as for sphere packing, quantum theory, crystallography, kissing numbers, etc. Nebe and Sloane made a list of famous lattices (mostly non-crypto related) available online at `http://www.math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/`: its huge size is a testimony of the popularity of lattices. The first book that comes to mind when mentioning the origins of lattice studies is the book of Minkowski [Min96]. We only provide in this section the basic knowledge needed to understand the next chapters (focusing on cryptography and number theory). Note that Micciancio is providing a free small course online at `https://cseweb.ucsd.edu/classes/sp14/cse206A-a/index.html` which is more "complete" than this chapter.

2.3.1 Lattice, volume and dual

Definition 5 (Lattice)**.**
We call lattice a discrete subgroup of $\mathbb{R}^n$ where n is a positive integer. We say a lattice is an integral lattice when it is a subgroup of $\mathbb{Z}^n$. A basis of the lattice is a basis as a $\mathbb{Z}-module$. If M is a matrix, we define $\mathcal{L}(M)$ the lattice generated by the rows of M.

The i-th row of a matrix M will be denoted M_i, and the element at the i-th row and j-th column $M_{i,j}$. We can also see a lattice by the set of integral combinations of a set of vectors $v_1, ...v_n$. In that case the lattice generated is noted $< v_1, .., v_n >_{\mathbb{Z}}$. This notation is useful when specific vectors from different sets are chosen rather than a single matrix. Note that we consider row vectors in this thesis, just like in coding theory.

In our work we only consider full-rank integral lattices (unless specified otherwise), i.e such that their basis can be represented by a $n \times n$ non-singular integral matrix.

It is important to note that just like in classical linear algebra, a lattice has an infinity of basis (assuming $n \geq 2$). In fact, if $\boldsymbol{B}$ if a basis of $\mathcal{L}$, then so is $\boldsymbol{UB}$ for any unimodular matrix $\boldsymbol{U}$ ($\boldsymbol{U}$ can be seen as the set of linear operations over $\mathbb{Z}^n$ on the rows of $\boldsymbol{B}$ that do not affect the determinant). Figure 2.1 and 2.2 shows an example of two different basis of the same lattice. We distinguish them with the adjectives "good" and "bad" for now, referring to the "capacity" a basis has to solve certain computational problems we will define later. Note that one basis is intuitively easier to use to determine all lattice points.

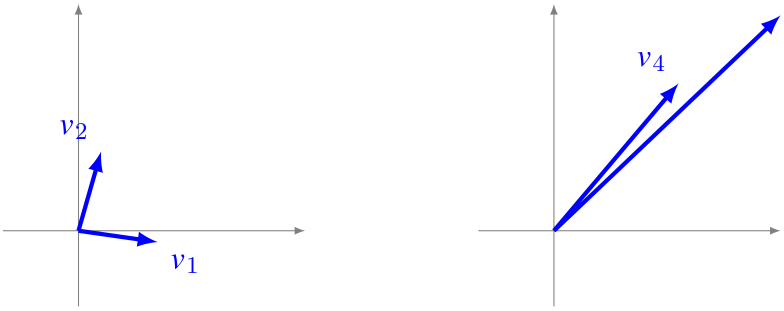


Figure 2.1: Two basis. Same lattices?

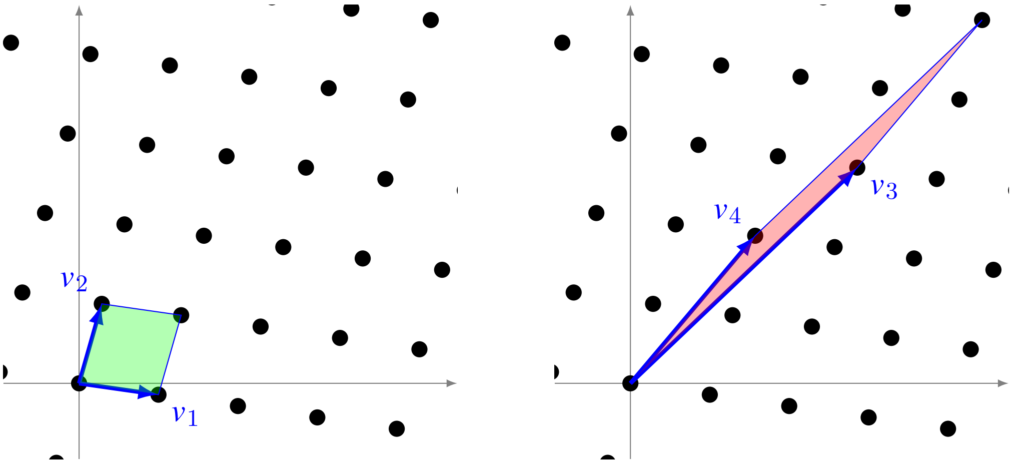


Figure 2.2: "Good" basis on left, "Bad" basis on right

Definition 6 (Inner Product).
Given a field $\mathcal{K}$, we say a bilinear application $\langle \mathcal{K}^n, \mathcal{K}^n \rangle \longrightarrow \mathcal{R}$ is a inner product, which is conjugate symmetric and positive-definite.

The "usual" inner product over $\mathbb{R}^n$ is the operation $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$. It is sometimes called dot product and is written $x \cdot y$.

Definition 7 (Orthogonality).
Given a ring $\mathcal{R}$, we say $x, y \in \mathbb{R}^n$ are orthogonal when $\langle x, y \rangle = 0$.

Definition 8 (Dual lattice).
Let $\mathcal{L}$ be an integral lattice. The dual lattice $\mathcal{L}$ is noted $\mathcal{L}^*$ and is defined as

$$\mathcal{L}^* = \{x \in \text{span}_{\mathbb{R}}(\mathcal{L}) : \forall y \in \mathcal{L},\ \langle x, y\rangle \in \mathbb{Z}\}$$

If B is a basis of $\mathcal{L}$, then a basis of $\mathcal{L}^*$ is given by $(BB^\top)^{-1}B$ and $(\mathcal{L}^*)^* = \mathcal{L}$ (where $B^\top$ is the transpose of B).

Note that $\mathcal{L}^*$ is in general not integral, and because we work with full-rank integral lattices only, we have $(B^\top)^{-1} = (B^{-1})^\top$.

Theorem 1 (Determinant)**.**
*For any lattice $\mathcal{L}$, there exists a real value we call **determinant**, denoted* $\det(\mathcal{L})$*, such that for any basis B,* $\det(\mathcal{L}) = \sqrt{\det(BB^\top)}$.

While we mention a "real value", the determinant of a full-rank matrix is within the same ring of the matrix elements, i.e, if we have a full-rank integral matrix, its determinant will necessary also be an integer. The literature sometimes call $det(\mathcal{L})$ the **volume** of $\mathcal{L}$ [Min96].

2.3.2 Hermite Normal Form and sublattices

We mentioned earlier that we have an infinite amount of basis, and in figure 2.2 both surfaces occupies the same amount of space. For any basis B of $\mathcal{L}$, we have $\det \leq \prod_1^n B_i$ where the equality holds if and only if all B_i are orthogonal with each other. One of the basis form we commonly use is the Hermite Normal Form (**HNF**), which showcases its determinant as the product of its diagonal elements (on the full-rank case).

Definition 9 (**HNF**)**.**
*Let $\mathcal{L}$ be an integral lattice of dimension d and $H \in \mathbb{Z}^{d,n}$ a basis of $\mathcal{L}$. H is said to be of **HNF** if and only if*

$$\forall 1 \leq i, j \leq d,\ H_{i,j} \begin{cases} = 0 & if\ i > j \\ \geq 0 & if\ i \leq j \\ < H_{j,j} & if\ i < j \end{cases}$$

The **HNF** can be computed from any basis in polynomial time [KB79], is unique [Coh93] per lattice and has a very compressed form, and thus is an ideal form for public keys [Mic01]. We denote **HNF**(M) the **HNF** of a matrix M. As the **HNF** form is unique per lattice, we can write **HNF**(M) = **HNF**($\mathcal{L}$).
One of the easiest form to work with when the **HNF** is computed is when we obtain a "perfect" **HNF**. Note that for the formal calculus software MAGMA [BCP97], the **HNF** have the index i, j reversed, leading to a triangular superior matrix instead of a triangular inferior matrix in the full-rank case. A small trick for users of MAGMA wanting a triangular inferior form is to use a central symmetry on the coefficients once before and once after the computation.

Definition 10 (Perfect **HNF**).
*We say **HNF**$(\mathcal{L})$ has pseudo-perfect form when only one column differs from Id_n and perfect when only the first column differs.*

Example 1. *A has perfect form, B pseudo-perfect, and C neither of them.*

$$A = \begin{bmatrix} 34 & 0 & 0 & 0 \\ 27 & 1 & 0 & 0 \\ 32 & 0 & 1 & 0 \\ 13 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 34 & 0 & 0 \\ 0 & 25 & 1 & 0 \\ 0 & 18 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \end{bmatrix}$$

Note that Goldstein and Mayer proved that the largest part of all lattices given a fixed dimension and determinant admits a perfect form [GM03] (for a large enough dimension and determinant). This form is also ideal for evaluating densities of certain lattice types assuming their **HNF** is known [BL09]. For consistency in the rest of the paper, we assume Id_n is a perfect form **HNF**.

This form is also probably the best form to use if we want to test the membership of a vector x inside a lattice $\mathcal{L}$, as any $x \in \mathcal{L}$ reduce to 0 via Gaussian Elimination (or Gauss-Jordan) [Coh93]. This property was used by Micciancio for lattice-based cryptography in [Mic01].

Example 2. *Is $v = [1 \; -1 \; 1 \; 1] \in \mathcal{L}(C)$?*

$$\left[\begin{array}{cccc} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \\ \hline 1 & -1 & 1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cccc} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \\ \hline -12 & -1 & 1 & 0 \end{array}\right] \rightarrow \left[\begin{array}{cccc} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \\ \hline -27 & -2 & 0 & 0 \end{array}\right] \rightarrow \left[\begin{array}{cccc} 17 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 \\ 15 & 1 & 1 & 0 \\ 13 & 0 & 0 & 1 \\ \hline -17 & 0 & 0 & 0 \end{array}\right]$$

The final step is easy to guess (reduce to 0) so the answer is yes.

Note that this reduction technique also allows to state some basic properties. For example, we can easily observe that for any $v \in \mathbb{Z}^n$, $(\det(\mathcal{L}) \times v) \in \mathcal{L}$. This property is extensively used in Chapter 3. A MAGMA code for this lattice membership test can be found in the appendix in figure A.1.

Definition 11 (Overlattice and sublattice).
We say a lattice $\mathcal{L}_1$ is an overlattice of $\mathcal{L}_2$ and $\mathcal{L}_2$ is a sublattice of $\mathcal{L}_1$ when $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

Figure 2.3 shows an example of an overlattice and a sublattice. Note that the lattice covering the bigger volume has actually less points, and is the sublattice. In particular, all full-rank integral lattices of dimension n are sublattices of $\mathbb{Z}^n$ (which

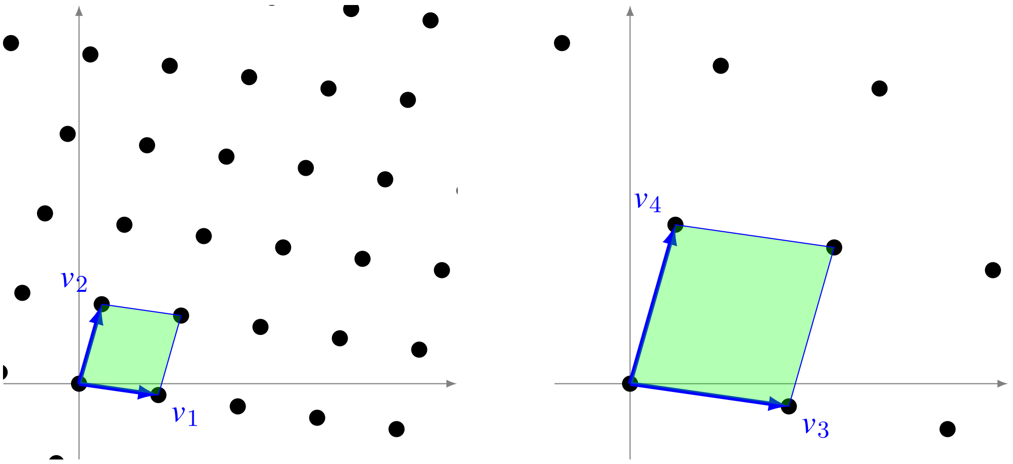


Figure 2.3: Overlattice $\mathcal{L}_1$ on left, Sublattice $\mathcal{L}_2$ on right, $\mathcal{L}_2 \subset \mathcal{L}_1$

has the identity matrix as a basis).

Note that this relationship is not limited to lattices of the same rank. It is very easy to compute sublattices of a lower rank (not full rank) as figure 2.4 show two sublattices of the lattices of figure 2.3.

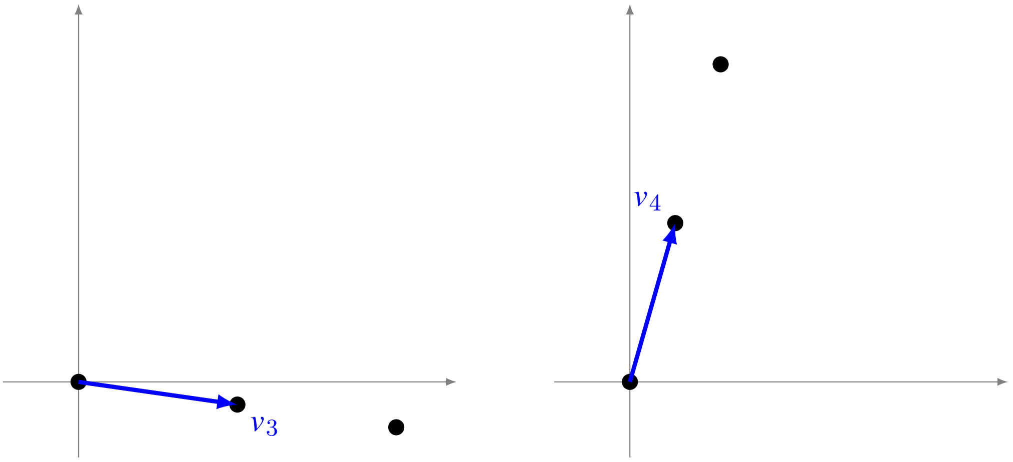


Figure 2.4: Two sublattices of $\mathcal{L}_1$ and $\mathcal{L}_2$ from figure 2.3

In any case, note that given a full-rank lattice, a full-rank sublattice has the bigger determinant and a full-rank overlattice has the smaller determinant. It is not necessarily true in other cases but as we stated earlier, we mostly work in the full-rank case.

2.3.3 Norms, lattice constants and heuristics

As many problems arising in lattice theory are about distances, and we define the distance of two points as the norm (or length) of their difference.

Definition 12 (Norms).
We say a function $N : \mathbb{R}^n \rightarrow \mathbb{R}$ is a norm on $\mathbb{R}^n$ when

- $N(x) > 0$ *for all non-zero* $x \in \mathbb{R}^n$*, and* $N(0) = 0$.
- $N(c \cdot x) = |c| \cdot N(x)$ *for all* $c \in \mathcal{R}^n$, $x \in \mathbb{R}^n$.
- $N(x + y) \leq N(x) + N(y)$ *for all* $x, y \in \mathbb{R}^n$.

Given an implicit norm we tend to note $\|x\|$ the value $N(x)$. The most popular norms in lattice theory are the max norm $\|(x_1, ..., x_n)\|_\infty = max_i \ x_i$ and the l_p norm $\|(x_1, ..., x_n)\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ for $p \geq 1$. The norm we use the most, the Euclidean norm, is the l_2 norm. See Figure 2.5 for examples. As we sometimes have to switch norms in order to solve equations involving multiple norms, it must be noted that for any couple of norms $(\|\cdot\|_\alpha, \|\cdot\|_\beta)$, there exists a constant $K_{\alpha,\beta}$ such that $\|\cdot\|_\alpha \leq K_{\alpha,\beta}\|\cdot\|_\beta$ (this is true for any finite-dimensional complex vector space). Those constants usually depends of the dimension, for example $K_{2,\infty} = \sqrt{n}$ and $K_{\infty,2} = 1$.

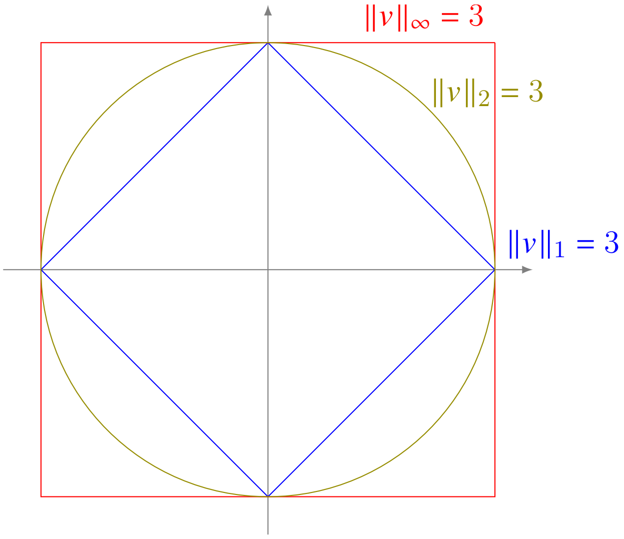


Figure 2.5: Points for $\|v\|_1 = 3$, $\|v\|_2 = 3$ and $\|v\|_\infty = 3$

Now we will define important constants that are often repeatedly used in the field.

Definition 13 (Minima).
We note $\lambda_i(\mathcal{L})$ the i−th minimum of a lattice $\mathcal{L}$. It is the radius of the smallest zero-centered ball containing at least i linearly independant elements of $\mathcal{L}$.

Note that usually the norm is known from context. In most cases, l_2 is implicit but if the minima is defined for another norm, let's say l_∞, the notation becomes more precise as $\lambda_i^\infty(\mathcal{L})$ or $\lambda_{i,\infty}(\mathcal{L})$.

Definition 14 (Hermite Constant)**.**
The Hermite Constant γ_n is defined as the biggest value attainable by $\lambda_1(\mathcal{L})^2$ for any given full-rank lattice $\mathcal{L}$ of dimension n.

The reason for the square is because Charles Hermite used to study quadratic forms. Therefore we are mostly interested in $\sqrt{\gamma_n}$. It is known that for all dimensions $n > 2$, we have $\gamma_n \le (\frac{4}{3})^{\frac{d-1}{2}}$. The exact value of the Hermite Constant is known only for dimensions 1 to 8 and 24, and is an open problem for other values.

Definition 15 (Lattice gap)**.**
We note $\delta_i(\mathcal{L})$ the ratio $\frac{\lambda_{i+1}(\mathcal{L})}{\lambda_i(\mathcal{L})}$ and call that a lattice gap. When mentioned without index and called "the" gap, the index is implied to be $i = 1$.

We also define the "root lattice gap", i.e elevated to the power $\frac{1}{n}$ where n is the dimension of the lattice. Note that, unlike the Hermite Constant, the lattice gap can have any value within $[1, \det(\mathcal{L})]$.

Next, we present Minkowski's theorems which give bounds in order to find lattice vectors.

Theorem 2 (Minkowski's Convex Body Theorem [Min96])**.**
Given a full rank lattice $\mathcal{L} \subset \mathbb{R}^n$, and a convex $S \subset \mathbb{R}^n$ that is symmetric with respect to the origin, then if $vol(S) > 2^n det(\mathcal{L})$, then there exists $x \neq 0^n$, such that $x \in S \cap \mathcal{L}$.

One typically used volume is the ball centered in the origin for S. Or one can directly use the bound from Blichfeldt $\gamma_n \le 1 + \frac{n}{4}$ [Bli14].

Theorem 3 (Minkowsi's second theorem [Min96])**.**
Let $\mathcal{L}$ be a full-rank lattice of dimension n, then

$$\left(\prod_{i=1}^{n} \lambda_n(\mathcal{L})\right)^{1/n} \le \sqrt{n} det(\mathcal{L})^{1/n}$$

Property 1 (Gaussian Heuristic (**GH**))**.**
Let $\mathcal{L} \subset \mathbb{Z}^n$ be a lattice and let $K \subset \mathbb{R}^n$ be a measurable subset of the real space. Then $|\mathcal{L} \cap K| \approx \frac{vol(K)}{vol(\mathcal{L})}$.

This allows us to approximate $\lambda_1(\mathcal{L})$ using a n-dimensional ball whose volume is approximately $det(\mathcal{L})$. Knowing that the volume of a n-dimensional ball of radius r is $V_n(r) \approx \frac{1}{\sqrt{n\pi}}(\frac{2\pi e}{n})^{n/2} r^n$, the end result is the following:

Property 2 (Second **GH**)**.**
For a random lattice $\mathcal{L}$ of dimension n, $\lambda_1(\mathcal{L}) \approx Det(\mathcal{L})^{1/n}\sqrt{\frac{n}{2\pi e}}$

The heuristic has been tested experimentally [GNR10], as well as in the context of lattice reduction [CN11][GN08] and has been found to be quite accurate starting from dimension $n > 45$ as far as random lattices are concerned. We can also note that if we use the Minkowski's first theorem, we can get an upper bound $\lambda_1(\mathcal{L}) \leq Det(\mathcal{L})^{1/n}\sqrt{\frac{2n}{\pi e}}$ which is only twice as large as the approximation obtained by the Gaussian Heuristic. We note that other competitive approximations for random lattices [Söd11, Kim16] do exist.

The next property is a work of [NS15].

Definition 16 (Density of lattices families).
As the dimension n grows, the number of co-cyclic lattices (i.e such that $\mathbb{Z}^n/\mathcal{L}$ is cyclic) converge to 85% and lattices with square-free determinant to 71.7%.

Given the **HNF** of $\mathcal{L}$, representing the group $\mathbb{Z}^n/\mathcal{L}$ is trivial. As a matter of fact, basic results can be easily deduced as the following two: "If B is a full-rank perfect **HNF**, then $\mathbb{Z}^n/\mathcal{L}(B)$ is co-cyclic" and "If $\det(B)$ is square-free then $\mathbb{Z}^n/\mathcal{L}(B)$ is co-cyclic" however both converse are false.

2.4 Computationally hard lattice problems

2.4.1 Shortest Vector Problem and variants

As we gave some of definitions and properties related to the length of the vectors, one of the most obvious problems is to find the shortest vector of a lattice.

Definition 17 (Shortest Vector Problem (**SVP**)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n, find $v \in \mathcal{L}$ such that $\|v\| = \lambda_1(B)$.

The problem was shown to be NP-hard for infinity norm in 1981 by Emde-Boas [vEB81] and then for l_2 under randomized reductions in 1998 by Ajtai [Ajt98]. As far as we know there is always a gap between random instances of NP-hard problems and specialized instances used in cryptography, and that gap is usually due to what we use as a trapdoor. Usually in cryptography, one does not need to find the shortest vector (actually it might not be unique, but a vector whose norm is the smallest) but a close approximation. How close that approximation can be in practice is one of the central points of research in lattice-based cryptanalysis. Hence the following definition of the problem's approximation:

Definition 18 (γ-approximate Shortest Vector Problem ($\mathbf{SVP}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n and an approximation factor γ, find $y \in \mathcal{L}$ such that $\|y\| \leq \gamma\lambda_1(B)$.

Figure 2.6 illustrates the difference between **SVP** and $\mathbf{SVP}_\gamma$ in the lattice $\mathcal{L} = \mathcal{L}(\{a, b\})$, i.e finding $y \in \mathcal{L}$ s.t $\|y\| = \lambda_1$ or $\|y\| \leq \gamma\lambda_1$. The green zones show the area of the valid solutions and the red dots the actual solutions.

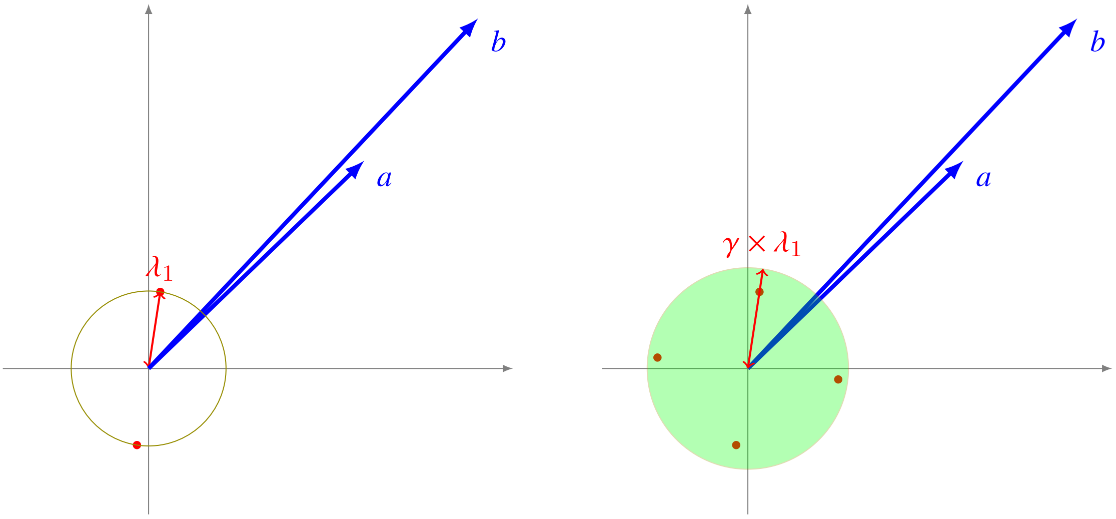


Figure 2.6: SVP on the left, $\text{SVP}_{1.3}$ on the right

Note that in l_2 under randomized reduction, the problem is NP-hard for any constant approximation factor [Kho05]. In l_∞, the problem is still NP-hard under deterministic reductions for quasi-polynomial approximation factors [Din00]. On the other hand, the problem is known to be in co-NP for an approximation factor of $\sqrt{n}$ and in co-AM for $\sqrt{n/\log n}$ [AR05, GG00].
However,it is still unknown if the problem is hard when γ is within polynomial factors, which is the case in almost all cryptographic applications. Furthermore, we tend to use specific lattices in cryptography, as we want to have a trapdoor for decryption, therefore proving that "breaking" a cryptosystem is as hard as solving $\mathbf{SVP}_\gamma$ within polynomial factors is far stretched. And as far as decoding goes, it is often the case that we only want a ciphertext to have one unique meaning when deciphered and not multiple ones, which give use us the following problem:

Definition 19 (γ-unique Shortest Vector Problem ($\mathbf{uSVP}_\gamma$)).
Given a basis of a lattice $\mathcal{L}$ with its lattice gap $\delta > 1$, solve ***SVP****.*

The condition $\delta > 1$ forces the shortest vector to be unique (with respect to the sign), which is not the case for the lattice showed in figure 2.10 which showcase an example with orthogonal basis vectors. Some cryptosystems are based on worst-case hardness on $\mathbf{uSVP}_\gamma$ with polynomial gap as [AD97] and [Reg04]. There exists also attacks that was specifically built to exploit high gaps [LWXZ11].

One of the main issues of all those **SVP** based problems is that given a randomly selected lattice $\mathcal{L}$, the value $\lambda_1(\mathcal{L})$ is hard to determine. Hence the following problem:

Definition 20 (γ-approximate Shortest Length Problem ($\mathbf{SLP}_\gamma$)).
Given a basis B of $\mathcal{L}$ and an approximation factor γ, find λ such that $\lambda_1(B) \leq \lambda \leq \gamma\lambda_1(B)$.

And the following decision problem:

Definition 21 (Decision Shortest Vector Problem (**DSVP**)).
Given a basis B of $\mathcal{L}$ of dimension n, and a real number $r > 0$, decide if $\lambda_1(\mathcal{L}) \leq r$.

Figure 2.7 shows the valid λ for $\mathbf{SLP}_\gamma$ and the correct "no" answers to DSVP.

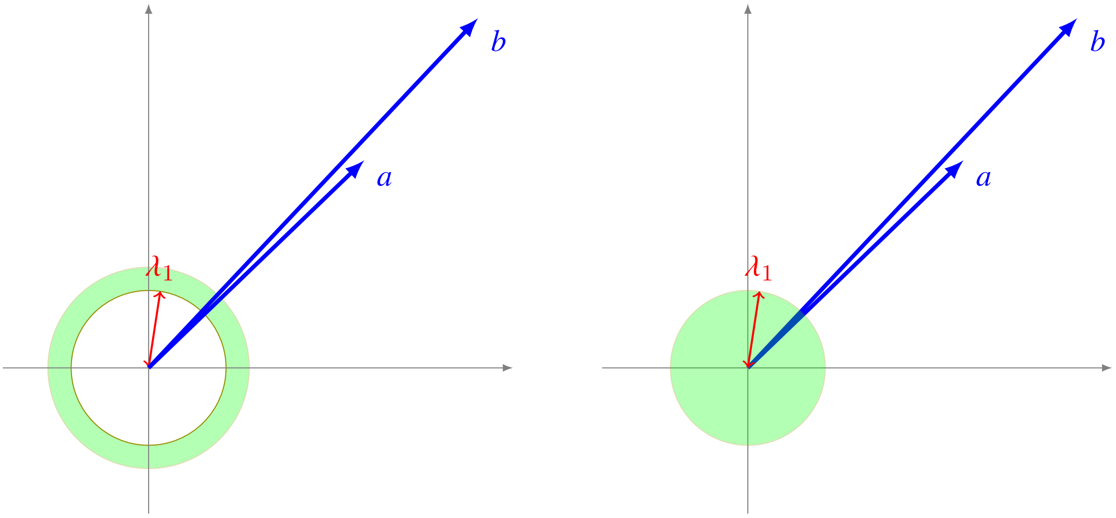


Figure 2.7: $\text{SLP}_{1.3}$ on the left, dSVP on the right

Another close problem is the following

Definition 22 (γ-Gap Shortest Vector Problem ($\mathbf{GapSVP}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n, an approximation factor γ and $r > 0$, decide whether $\lambda_1(\mathcal{L}) \leq r$ or $\lambda_1(\mathcal{L}) \geq \gamma r$.

This problem is NP-hard for constant δ [Kho05]. Since $\lambda_1(\mathcal{L})$ is basically unknown to us in general, it is very tricky to measure the efficiency of an algorithm. Therefore, to measure algorithm efficiency we must be able to define a problem which variables can be easily computed:

Definition 23 (γ-Hermite Shortest Vector Problem ($\mathbf{HSVP}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n and a factor γ we call Hermite Factor, find $y \in \mathcal{L}$ such that $\|y\| \leq \gamma det(\mathcal{L})^{1/n}$.

The definition of the Hermite Factor is obviously linked to the Hermite Constant, as the solution's existence is guaranteed as soon as $\gamma \geq \sqrt{\gamma_n}$ where γ_n is the Hermite Constant for full rank lattices of dimension n. We can also see that an algorithm that solves $\mathbf{SVP}_\gamma$ solves $\mathbf{HSVP}_{\gamma\sqrt{\gamma_n}}$ since $\gamma\lambda_1(\mathcal{L}) \leq \gamma\sqrt{\gamma_n}det(\mathcal{L})^{1/n}$.

2.4.2 Closest Vector Problem and variants

The other famous problem on lattices is the Closest Vector Problem (**CVP**).

Definition 24 (**CVP**).
Given a basis B of a lattice $\mathcal{L}$ of dimension n and $t \in \mathbb{R}^n$, find $y \in \mathcal{L}$ such that $\forall y_2 \in \mathcal{L}, \|t - y\| \leq \|t - y_2\|$.

CVP is also known to be NP-hard [vEB81]. As there exists subproblems for **SVP**, the same exists for **CVP**:

Definition 25 (γ-Approximate Closest Vector Problem ($\mathbf{CVP}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n, an approximation factor γ and $t \in \mathbb{R}^n$, find $y \in \mathcal{L}$ such that $\forall y_2 \in \mathcal{L}, \|t - y\| \leq \gamma\|t - y_2\|$.

$\mathbf{CVP}_\gamma$ is known to be NP-hard for any constant γ, and even for sub-polynomial and quasi-linear γ (in the dimension) [ABSS93]. Furthermore, $\mathbf{CVP}_\gamma$ is at least as hard as $\mathbf{SVP}_\gamma$ [GMSS99]. Similar to the previous figure for **SVP**, figure 2.8 shows examples solutions for **CVP** and $\mathbf{CVP}_\gamma$: the first one has one solution (no point can be closer) and the second has two (two points verify other points cannot be much closer). From a certain point of view, solving **CVP** can be seen as solving **SVP** with a lattice $\mathcal{L}$ translated by the target vector $t \notin \mathcal{L}$ (which would then not be a lattice anymore).

A close problem to the **CVP** is the Bounded Distance Decoding (**BDD**).

Definition 26 (γ-Bounded Distance Decoding ($\mathbf{BDD}_\gamma$)).
*Given a basis B of a lattice $\mathcal{L}$, a point x and a approximation factor γ ensuring $d(x, \mathcal{L}) < \gamma\lambda_1(B)$ find the lattice vector $v \in \mathcal{L}$ closest to x. When $\gamma = 1/2$, we can omit γ and note the problem **BDD**.*

The main difference between $\mathbf{BDD}_\gamma$ and $\mathbf{CVP}_\gamma$ is on the the comparisons with other points: $\mathbf{BDD}_\gamma$'s solutions only need to be close enough to the target point

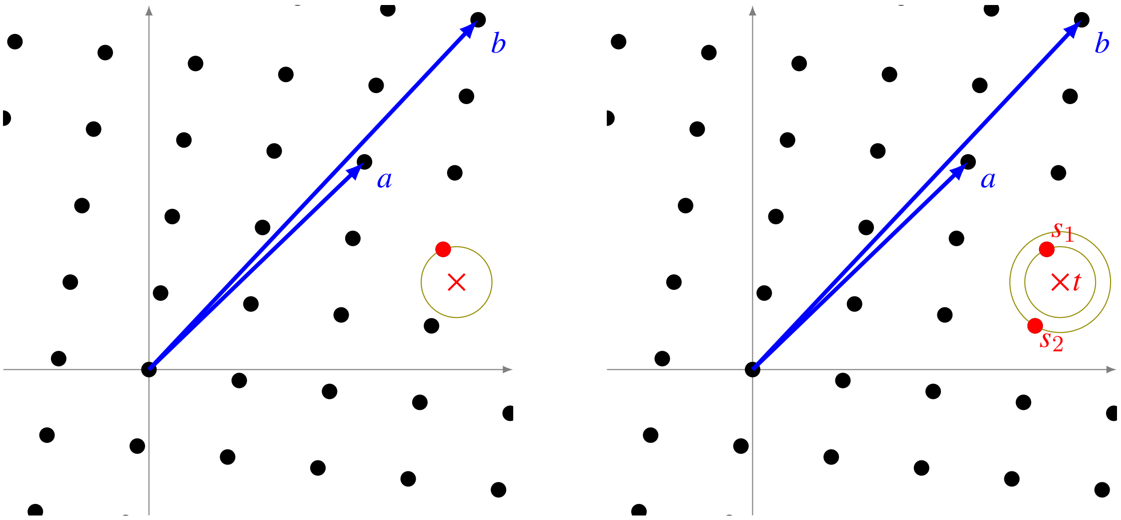


Figure 2.8: **CVP** on left, $\mathbf{CVP}_{1.5}$ on right: $\|t - s_2\| \leq 1.5 \times \|t - s_1\|$

regardless of the other points, while in $\mathbf{CVP}_\gamma$ the solutions need to be closer than the other points up to a factor γ. As pointed out in [LM09], this definition of the $\mathbf{BDD}_\gamma$ is actually harder as γ grows. It is also proved in the same paper that $\mathbf{BDD}_{1/(2\gamma)}$ reduces itself to $\mathbf{uSVP}_\gamma$ in polynomial time and the same goes from $\mathbf{uSVP}_\gamma$ to $\mathbf{BDD}_{1/\gamma}$ when γ is polynomially bounded by n, which is typically the case in cryptography thus solving one or the other is of equal interest to us. Therefore, it is common to see **BDD** referenced rather than $\mathbf{uSVP}_\gamma$ as **BDD** is a problem that is being well-studied in coding theory. It is known that $\mathbf{BDD}_\gamma$ is NP-hard for $\gamma > \sqrt[p]{2}^{-1}$ when using the l_p norm [LLM06].

Yet another close problem is the following:

Definition 27 (γ-Guaranteed Distance Decoding ($\mathbf{GDD}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$, and a bound $\gamma > \lambda_1(\mathcal{L})$, for any point x find a lattice vector $v \in \mathcal{L}$ such that $\|x - v\| < \gamma$.

The $\mathbf{GDD}_\gamma$ problem is simpler to state and less restrictive than the $\mathbf{BDD}_\gamma$: it is also easier to verify than the $\mathbf{BDD}_\gamma$ especially when the lattice minima λ_1 is unknown. See figure 2.9 which shows the decoding distance in green from each point: a point in an overlapping green area has multiple solutions. Note that points in white spaces have no solution whereas they would have a solution in **CVP**.

2.4.3 Shortest Basis Problem and variants

In lattice-based cryptography, some schemes relies on having a good basis as a secret key, and a bad basis as a public key (see Goldreich-Goldwasser-Halevi (**GGH**)

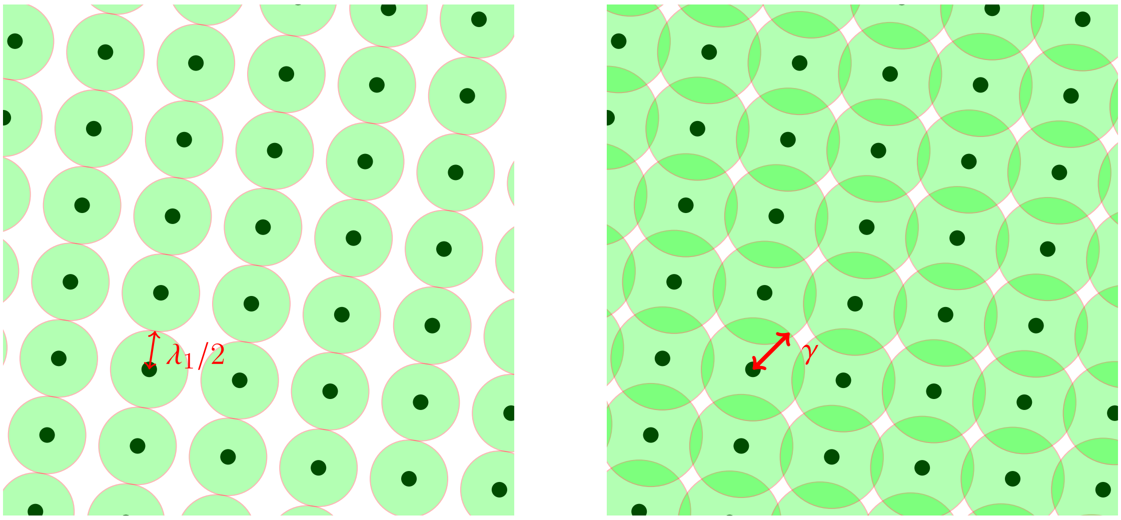


Figure 2.9: $\mathbf{BDD}_{1/2}$ on left, **$\mathbf{GDD}_\gamma$** on right

[GGH97] for example) in order to solve the previously stated problems. By generalizing the notion of solving **SVP**, i.e finding the first minima, one can define the problem of finding successive minimas, which lead to finding "good" basis or at least independent vectors.

Definition 28 (Successive Minima Problem (**SMP**)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n, find a set of linearly independant vectors $y_1, ..., y_n \in \mathcal{L}$ such that $\|y_i\| = \lambda_i(B)$ for $1 \leq i \leq n$.

The problem is obviously at least as hard as **SVP** and can also be relaxed by an approximation factor:

Definition 29 (γ-Approximate Successive Minima Problem (**SMP$_\gamma$**)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n and an approximation factor γ, find a set of linearly independant vectors $y_1, ..., y_n \in \mathcal{L}$ such that $\|y_i\| \leq \gamma\lambda_i(B)$.

The basis version:

Definition 30 (Shortest Basis Problem (**SBP**)).
Given a basis of a lattice $\mathcal{L}$ of rank n, find a basis $y_1, ..., y_n \in \mathcal{L}$ such that

$$max_i\|y_i\| \leq min\{max_j\|a_j\| \ |\{a_1, ..., a_n\} \text{ is a basis of } \mathcal{L}\}.$$

and its relaxation:

Definition 31 (γ-Approximate Shortest Basis Problem (**SBP$_\gamma$**)).
Given a basis of a lattice $\mathcal{L}$ of rank n, find a basis $y_1, ..., y_n \in \mathcal{L}$ such that $max_i\|y_i\| \leq min\{max_j\|a_j\| \ |\{a_1, ..., a_n\}$ is a basis of $\mathcal{L}\}$.

Note that the basis version **SBP** and the vector version **SMP** are different: at dimension $n \geq 5$, a solution to **SMP** is not necessarily even a solution to **SBP**. The best way to illustrate this phenomenon is by using an example, which we will take from Antoine Joux's PhD thesis [Jou93] in fig 2.10.

SBP solution:
must form a basis of $\mathcal{L}$

$$B = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- $\lambda_1 = 2$
- $\lambda_2 = 2$
- $\lambda_3 = 2$
- $\lambda_4 = 2$
- $\lambda_5 = 2$

SMP solution:
forms a sublattice basis.

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Figure 2.10: An example of **SBP** $\neq$ **SMP** for $\mathcal{L} = \mathcal{L}(B)$

And another problem that is close to $\mathbf{SBP}_\gamma$ is the following:

Definition 32 (Shortest Independent Vector Problem (**SIVP**)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n, find a set of linearly independant vectors $y_1, ..., y_n \in \mathcal{L}$ such that $max_i \|y_i\| \leq \lambda_n(\mathcal{L})$.

Definition 33 (γ-Approximate Shortest Independent Vector Problem ($\mathbf{SIVP}_\gamma$)).
Given a basis B of a lattice $\mathcal{L}$ of dimension n and an approximation factor γ, find a set of linearly independant vectors $y_1, ..., y_n \in \mathcal{L}$ such that $max_i \|y_i\| \leq \gamma\lambda_n(\mathcal{L})$.

This problem is NP-hard whenever $\gamma \leq n^{1/\log\log n}$. [BS99]. $\mathbf{SBP}_\gamma$ and $\mathbf{SIVP}_\gamma$ are "close" in the sense that they reduce to each other within an approximation factor $\sqrt{n}/2$ [Mic08], and in the same work it is shown that both of them reduce themselves to $\mathbf{SVP}_\gamma$ within an approximation factor $\sqrt{n}$. Furthermore, any solution to **SMP** is a solution to **SIVP** [Mic08].

2.5 Algorithms to solve lattice problems

2.5.1 Gram-Schmidt Orthogonalization algorithm

Before we go on presenting some lattice reduction algorithms we must present the Gram-Schmidt Orthogonalization (**GSO**) algorithm (Alg 1). **GSO** applications are not limited to lattices. Given an integral basis, the algorithm outputs a fully-orthogonal matrix over the rational field $\mathbb{Q}$. However, **GSO** is a key tool in constructing lattice reduction algorithms.

Note that the result of the **GSO** will depend of the first vector b_1. Intuitively, the shorter b_1 is, the most likely the final basis will have short vectors although this is not guaranteed. Denoting $\frac{\langle b_i^*, b_j^* \rangle}{\|b_j^*\|^2} = g_{i,j}$, notice how all operations are represented in figure 2.11, giving us a relationship between the entry of the **GSO** process and

Algorithm 1 **GSO** algorithm

```
Require: a basis $B = \{b_1, .., b_n\}$ of $\mathcal{L}$
Ensure: a set of vectors $S^* = \{b_1^*, .., b_n^*\}$ with $b_1^* = b_1$ and $\langle b_i^*, b_j^* \rangle = 0$ for $i \neq j$
  $S^* \leftarrow S$
  for $i = 1$ to $n$ do $b_i^* \leftarrow b_i$
      for $j = 1$ to $i - 1$ do
          $b_i^* \leftarrow b_i^* - \frac{\langle b_i^*, b_j^* \rangle}{\|b_j^*\|^2} b_j^*$        ▷ Making $b_i^*$ orthogonal to $b_j^*$
  return $S^*$
```

its output $G \times B^* = B$ where G is called the *Gram-Schmidt transformation matrix.* Interesting point following is given $B^* = \{b_1^*, ..., b_n^*\}$ the result of the **GSO** on B, $det(\mathcal{L}(B)) = \prod_{i=1}^{n} \|b_i^*\|$.

The **GSO** algorithm	The reverse operation
$b_1^* = b_1$	$b_1 = b_1^*$
$b_2^* = b_2 - g_{2,1} b_1^*$	$b_2 = g_{2,1} b_1^* + b_2^*$
$b_3^* = b_3 - g_{3,1} b_1^* - g_{3,2} b_2^*$	$b_3 = g_{3,1} b_1^* + g_{3,2} b_2^* + b_3^*$
$b_4^* = b_4 - g_{4,1} b_1^* - g_{4,2} b_2^* - g_{4,3} b_3^*$	$b_4 = g_{4,1} b_1^* + g_{4,2} b_2^* + g_{4,3} b_3^* + b_4^*$
$\vdots$	$\vdots$
$b_n^* = b_n - \sum_{i=1}^{n-1} g_{n,i} b_i^*$	$b_n = \sum_{i=1}^{n-1} g_{n,i} b_i^* + b_n^*$

$$B = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ g_{2,1} & 1 & 0 & \dots & 0 \\ g_{3,1} & g_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & g_{n,3} & \dots & 1 \end{bmatrix} \times B^*$$

Figure 2.11: General Gram-Schmidt process

2.5.2 Babai's algorithms

To solve lattice problems, we noticed earlier the better the basis the easier it becomes (see figure 2.2). Using a "good" basis (preferably with "short" and "relatively" orthogonal vectors), Babai's introduced two methods [Bab86] to solve **CVP**. The first is the *rounding-off* algorithm (Alg 2).

Algorithm 2 Babai's Rounding Off Algorithm

```
Require: $v \in \mathbb{Z}^n$, $B \in \mathbb{Z}^{n \times n}$ a basis of $\mathcal{L}$
Ensure: $w \in \mathcal{L}$ a close vector of $v$
  return $\lfloor v \times B^{-1} \rceil \times B$
```

The second one is the *nearest-plane* algorithm (Alg 3). Let us denote $B^* = b_1^*, ..., b_k^*$ the result of the **GSO** algorithm on $B = b_1, ..., b_k$.

Algorithm 3 Babai's Nearest Plane Algorithm

```
Require: $v \in \mathbb{Z}^n$, $B \in \mathbb{Z}^{m \times n}$ a basis of $\mathcal{L}$, $b_i^*$ GSO coefficients of $B$
Ensure: $w \in \mathcal{L}$ a close vector of $v$
$t \leftarrow v$
for $i = m$ down to 1 do
    $t \leftarrow t - \lfloor \frac{\langle t, b_i^* \rangle}{\langle b_i^*, b_i^* \rangle} \rceil b_i$        ▷ Make $t$ more orthogonal to $b_i$
return $v - t$
```

Babai's nearest plane algorithm solve $\mathbf{CVP}_\gamma$ in polynomial time for $\gamma = 2(\frac{4}{3})^{n/2}$.

Both algorithms output quality depends of the basis given as input. Given a very "good" basis (let's say a secret key), those algorithms can serve as a decryption method. However there is a crucial problem: how do we obtain a good basis? We present below lattice reduction algorithms to obtain "good" basis from "bad" ones, which makes use of the **GSO** algorithm.

2.5.3 Size-Reduction

The **GSO** transformation does not give a basis of a lattice. To keep the transformed basis, we need to perform integral transformations. Therefore the first idea is to apply the same operations as the **GSO** algorithm, but by rounding $g_{i,j}$ by its closest value. By doing so, we can also ensure the basis is "close" to a regular rational **GSO** output and hoping the coefficients of the **GSO** transformation matrix will be close as close to possible to 0. Basically, we want to transform B to B' such that the equality $B = G \times B^*$ becomes $B' = G' \times B^*$ and $|g'_{i,j}| \leq 1/2$.

However there is slight problem: everytime we update a vector of B (and so G), we also potentially change B^*. Therefore we need to proceed in a way that leaves B^* unchanged. Algorithm 4 does exactly what we want. Note that the input/output of G/G' is optional, as they can be recomputed.

2.5.4 Gauss's algorithm, the case of dimension 2

Before we describe the first relevant lattice reduction algorithm, we first give here a description of its ancestor. Gauss algorithm (alg 5) solve the **SVP** (and the **SIVP**, **SBP**, ...) in dimension 2.

Note that the basic idea is the same as the size-reduction (Alg 4): orthogonalize but using only integral linear combinations. The novelty here is that *vectors are swapped after orthogonalization*, unlike previous algorithms where the first vector was left untouched, and the algorithm is *rerun until the first vector cannot be short-*

Algorithm 4 Size-reduce

```
Require: a basis $B = \{b_1, .., b_n\}$ of $\mathcal{L}$, the matrix $G$ containg all $g_{i,j}$
Ensure: a basis $B' = \{b'_1, .., b'_n\}$ of $\mathcal{L}$ with $b'_1 = b_1$ and $|g'_{i,j}| \leq 1/2$ for $i \neq j$
$B' \leftarrow B$
$G' \leftarrow G$
for $i = 2$ to $n$ do
    for $j = i - 1$ down to 1 do
        $b'_i \leftarrow b'_i - \lfloor g'_{i,j} \rceil b'_j$    ▷ Backwards orthogonalization of $b_i$ from $b_{i-1}$ to $b_1$
        for $k = 1$ to $j$ do
            $g'_{i,k} \leftarrow g'_{i,k} - \lfloor g'_{i,j} \rceil g'_{j,k}$    ▷ Update $G'$'s $k$-th column
return $B'$,$G'$
```

Algorithm 5 Gauss Reduction Algorithm

```
Require: $v, w \in \mathbb{Z}^n$ where $v, w$ is a basis of $\mathcal{L}$
Ensure: $v, w$ is a short basis of $\mathcal{L}$
if $\|w\| < \|v\|$ then Swap $v, w$
while $\|v\| < \|w\|$ do
    $w \leftarrow w - \lfloor \frac{\langle w,v \rangle}{\|v\|^2} \rceil \times v$
    Swap $v, w$
return $v, w$
```

ened. This ensured that the first vector was effectively the shortest one. For more general results on Gauss' algorithm, we refer the readers to the work of Daudé, Flajolet and Vallée [DFV97] and subsequent works. In a sense, the Greatest Common Divisor (GCD) algorithm is solving the problem in dimension 1 and Gauss is an extension to dimension 2. Generalizing in higher dimensions however, is not trivial, and the next algorithm is probably its most famous attempt.

2.5.5 Lenstra-Lenstra-Lovász basis reduction algorithm

This algorithm due to Lenstra, Lenstra and Lovász [LLL82] was the first major lattice reduction algorithm for generic dimensions: it can be considered as a generalization of Gauss' algorithm for non-2 dimensions, Lenstra-Lenstra-Lovász basis reduction algorithm (**LLL**) uses the **GSO**, and swaps vectors to make shorter vectors at the top and retry. **LLL** is, however, much more tricky to understand, and was initially proposed to factorize polynomials over $\mathbb{Q}$ before its usage in cryptography. First of all, we need another testing condition to do better than *size-reduce*. Such a condition can is provided by the *Lovász condition*. Using the same notations as before, let us denote G such that $G \times B^* = B$ (see figure 2.11). Before we present the algorithm we define the *Lovász condition* for a given δ and $i \in [2, n]$ and a basis B of rank n:

$$\frac{\|b_i^*\|^2}{\|b_{i-1}^*\|^2} \geq \delta - g_{i,i-1}^2$$

Note that by fixing $\delta = 1$, we can find back Gauss' algorithm for dimension 2.

Everytime $\boldsymbol{B}$ is changed in the following algorithm (Alg 6), so is $\boldsymbol{B}^*$ and $\boldsymbol{G}$ to make the equality remain true. Every change is computed in polynomial time, therefore we will not write them explicitly in the algorithm description:

Algorithm 6 LLL

```
Require: $B$ a basis of $\mathcal{L}$ of dimension $n$, and a constant $\delta \in [\frac{1}{4}, 1]$
Ensure: $B$ a reduced basis of $\mathcal{L}$ satisfying the Lovász condition
i ← 2
while $i \leq n$ do
    $b_i \leftarrow b_i - \sum_{j=1}^{i-1} \lfloor g_{j,i} \rceil b_j$        ▷ (update the GSO)
    if Lovász condition is respected for $\delta, i$ then
        $i \leftarrow i + 1$
    else
        $swap(b_i, b_{i-1})$
        $i \leftarrow max\{2, i-1\}$
return $B$
```

Subsequent works modifying the original **LLL** have existed for various purposes. One is a blockwise generalization we will present later in this section, others modify the algorithm's swaps [FSW14, YY19], others focuses on data structures implementation to accelerate computation time [EJ16], find surprising links to other algorithms [LMHG13], or modify the conditions to achieve different reductions criteria [HMM98]. To summarize, the popularity of **LLL** goes way beyond cryptography. A first entry read to know more about the algorithm itself would be [NV10] or [NS09].

Like the the GCD algorithm or Gauss, **LLL** runs in polynomial time. Unlike the GCD algorithm or Gauss, **LLL** does not solve exactly **SVP** or **SBP**. However, given a full rank lattice of "large" dimension n, it does seem to solve the $\mathbf{HSVP}_\gamma$ with a Hermite factor of approximately 1.02^n [GN08], while the best proven upper bound is around 1.07^n.

For now, we will focus on the algorithms that are the most popular for cryptanalytic usages.

2.5.6 Enumeration techniques

Sometimes an approximate solution for a polynomial cost is not good enough. It is probable that some parties want to actually compute a correct solution at an exponential cost. Exhaustive search should do the job, but since there is an infinity of basis (therefore an infinity of vectors), we need to do exhaustive search in a proper way. The solution we present here is an exhaustive search, but an intelligent one: "*enumerations*".

Enumeration techniques trace back to as early as 1981 [Poh81], [Kan83] and [FP85]. The concept is to try every combination of basis vectors until we find the shortest vector, but by bounding the complexity by limiting the number of combinations in our exhaustive search. We also need to ensure that the correct result lies within the set bounds. Therefore one "easy" bound to limit is the size of the vectors we are searching into. Such a bound do exist according to Minkowski's Convex Body theorem (Theorem 2) and we can use the **GH** (Prop 1) to determine a bound R such that $\lambda_1(\mathcal{L}) \leq R$. Given a basis $B = (b_1, ..., b_n)$ of $\mathcal{L}$ such that vectors are sorted as $\|b_i\| \leq \|b_{i+1}\|$, we initially fix $R = \|b_1\|$.

The enumeration technique, like lattice reduction techniques, makes use of the **GSO**. Let $B^* = (b_1^*, ..., b_n^*)$ the result of the **GSO** algorithm applied to B (and note $G \times B^* = B$). Remember figure 2.11, and see that every basis vector can be represented by a sum of **GSO** vectors. If every basis vector can be represented by a set of **GSO** vectors, then so are the vectors themselves. Let us denote a vector $v \in \mathcal{L}$. Using figure 2.11 to obtain figure 2.12, we can write $v = \sum_{i=1}^{n}(v_i + \sum_{j=i+1}^{n} v_j g_{j,i})b_i^*$, and the projection over the k last **GSO** vectors:

$$P_k(v) = \textstyle\sum_{i=k}^{n}(v_i + \sum_{j=i+1}^{n} v_j g_{j,i})b_i^*$$

And by orthogonality of the **GSO** vectors:

$$\|P_k(v)\| = \textstyle\sum_{i=k}^{n} \|(v_i + \sum_{j=i+1}^{n} v_j g_{j,i})b_i^*\|$$

And it naturally follows that for any vector $v \in \mathcal{L}$

$$\|P_n(v)\| \leq \|P_{n-1}(v)\| \leq \cdots \leq \|P_1(v)\| = \|v\|$$

So, the enumeration technique consists in a bounded search within each projection level: first in $P_n(\mathcal{L})$, then in $P_{n-1}(\mathcal{L})$, etc. Searching in $P_k(\mathcal{L})$ given k consists of using all combinations possible of $v \in \mathcal{L}$ such that $P_n(\mathcal{L}) \leq R$. However this is going back to the initial problem: we need to determine how to choose v_i in $v = (v_1, ..., v_n)$.

We have bounds on the values v_i depending of the values obtained in v_j for $j > i$, therefore we must start from the first value v_n, and $\|v_n b_n^*\| \leq R$. Once we have a candidate for v_n, we must continue with v_{n-1} such that $\|v_n b_n^*\| + \|v_{n-1} b_{n-1}^*\| \leq R$, then v_{n-2}, etc. The bounds are not trivial, but every bounds for v_i given all v_j for $j > i$ depends uniquely of the values chosen for v_j in previous iterations, the **GSO** vectors and the coefficients of G, and the initially fixed radius R, i.e

$$F_-((v_{i+1}, ..., v_n), R, B^*, G) < v_i < F_+((v_{i+1}, ..., v_n), R, B^*, G)$$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}^T \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}^T \times \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ g_{2,1} & 1 & 0 & \dots & 0 \\ g_{3,1} & g_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & g_{n,3} & \dots & 1 \end{bmatrix} \times \begin{bmatrix} b_1^* \\ b_2^* \\ b_3^* \\ \vdots \\ b_n^* \end{bmatrix}$$

$$P_k(v) = \begin{bmatrix} v_1 \\ \vdots \\ v_{k-1} \\ v_k \\ \vdots \\ v_n \end{bmatrix}^T \times \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & 0 & 0 & \dots & 0 \\ g_{k-1,1} & \dots & 1 & 0 & \dots & 0 \\ g_{k,1} & \dots & g_{k,k-1} & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & \dots & g_{n,k-1} & g_{n,k} & \dots & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_k^* \\ \vdots \\ b_n^* \end{bmatrix}$$

$$P_k(v) = \begin{bmatrix} v_k \\ \vdots \\ v_n \end{bmatrix}^T \times \begin{bmatrix} 1 & 0 & \dots & 0 \\ g_{k+1,k} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,k} & \dots & g_{n,n-1} & 1 \end{bmatrix} \times \begin{bmatrix} b_k^* \\ \vdots \\ b_n^* \end{bmatrix}$$

Figure 2.12: Projection of a vector $v \in \mathcal{L}(B)$ over $\langle B^* \rangle$

Therefore we can see the enumeration algorithm as a search in a tree with n levels, each level becoming more restrictive in available branches depending on the parent branch. With that in mind, the algorithm can be written as described in Alg 7.

Algorithm 7 The Enumeration Algorithm for SVP

Require: B a basis of $\mathcal{L}$ of dimension n
Ensure: $v = \sum_i v_i b_i$ such that $v = \lambda_1(\mathcal{L}(B))$

```
while There is still unexplored nodes do
    if current node has $\|(0, ..., 0, v_i, .., v_n)\| < R$ then
        Go down in the tree, explore all possibilities for $v_{i-1}$
    else
        Remove the node, go up a level, search another node
return $B$
```

One can see that the size of the search tree grows exponentially depending on the bounds. Therefore, the lower is R and the better is the initial basis B, the better the the enumeration algorithm behave. Upgraded techniques for enumerations exists, such as pruning [GNR10], and alternative ways to solve **SVP** exists such as sieving [AKS01], using voronoi cells [MV13]... The point here is mostly to show solvers exists, and those are exploited as building blocks for further lattice reduction algorithms.

2.5.7 Hermite-Korkine-Zolotarev

Sometimes we do not only want the shortest vector, but we also want to have a very good basis along with it. Using an oracle $\mathcal{O}$ which solves **SVP** for up to k dimensions (examples have been provided in the previous subsection), we can try to achieve better.

First of all, we need a better condition than the *Lovász condition*. We present one here: the Korkine-Zolotarev (**KZ**) reduction criteria [KZ73], giving us the Hermite-Korkine-Zolotarev (**HKZ**) reduction algorithm. In a certain sense, a **KZ** reduced basis is the optimal reduction we can hope from a basis, as its first vector is equal to the shortest vector of the lattice. Noting $\pi_i(\mathcal{L})$ the projection of $\mathcal{L}$ over $\langle b_1, ..., b_{i-1}\rangle^\perp$, we say a basis $B = b_1, ..., b_k$ is **KZ** reduced if the following hold:

$$\forall i \in [1, d], \|b_i^*\| = \lambda_1(\pi_i(\mathcal{L}))$$

and this implies (without explaining the details):

$$\frac{\|b_1\|}{\lambda_1(\mathcal{L})} = 1, \frac{\|b_2\|^2}{\lambda_2^2(\mathcal{L})} \leq \frac{4}{3}, \frac{\|b_3\|^2}{\lambda_3^2(\mathcal{L})} \leq \frac{6}{4}$$
$$\forall i \geq 4, \frac{\|b_i\|^2}{\lambda_i^2(\mathcal{L})} \leq \frac{i+3}{4}$$

The **HKZ** basis reduction algorithm is then described in algorithm 8.

Algorithm 8 **HKZ** reduction algorithm

```
Require: $B = \{b_1, ...b_n\}$ a basis of $\mathcal{L}$ of dimension $n$
Ensure: $B$ a reduced basis of such that $\|b_i^*\| = \lambda_1(\pi_i(\mathcal{L}))$
Call $\mathcal{O}$ to find $b_1'$ of length $\lambda_1(\mathcal{L})$
Construct $B' = \{b_2', ..., b_n'\}$ where $\mathcal{L}(B') \neq \mathcal{L}$ but $\mathcal{L}(\{b_1'\} \cup B') = \mathcal{L}$
$B' \leftarrow$ HKZ($B'$)
$B \leftarrow \{b_1'\} \cup B'$
Call Size-reduce on $B$.
return $B$
```

Note that **HKZ** is sometimes abbreviated **KZ** (for the reduction criteria). It is unusable for high dimensions, but is however another building block for the next basis reduction algorithm.

2.5.8 Block Korkine-Zolotarev Reduction algorithm

Block Korkine-Zolotarev (**BKZ**) was first proposed by Schnorr and Euchner [SE94]. As the name implies, it uses the **HKZ** reduction algorithm as a subroutine in blocks. Suppose the oracle runs up to dimension k, and our basis has dimension $n > k$, then the **BKZ** algorithm is described in 9.

Algorithm 9 **BKZ** basis reduction algorithm

```
Require: $B = \{b_1, ...b_n\}$ a basis of $\mathcal{L}$, and a SVP-oracle up to dim $k < n$
Ensure: $B$ a reduced basis of such that $\|b_i^*\| = \lambda_1(\pi_i(\mathcal{L}))$
while changes occurs do
    for $i = 1$ to $n - k + 1$ do
        KZ-reduce the block $\pi_i(\{b_i, ..., b_{i+k-1}\})$ then lift it in $B$
        Use LLL on $B = \{b_1, ..., b_n\}$
return $B$
```

The literature will often mention the **BKZ** algorithm followed by a number, i.e BKZ-20, BKZ-30, BKZ-40... The number actually just represents the size k of the block where the **KZ** reduction is actually applied. In fact, we can see BKZ-2 as nothing more than **LLL**. Complexity estimations are given in [CN11]. Just like **LLL**, the efficiency in practice is not well understood: like **LLL** it behaves much better than the proven upper bounds. It does however improve the Hermite Factor given by **LLL**: in [GN08], the Hermite Factor of BKZ-20 is given as 1.0128, BKZ-28 gives 1.0109. It does seem hard to lower the Hermite Factor below 1.009 in reasonable time and it was claimed in the same paper that 1.005 for dimension 500 seems unreachable for random lattices. While lattice reduction techniques and their predictions had improved over the years (see [MW16, YD17, BSW18] and subsequent works), this claim does not seem to have been challenged so far.

2.5.9 Kannan's extension technique

We present here a technique that allows to transform instances of **CVP** to **SVP** or to solve **SVP** by expanding a lattice. Heuristically solving **SVP** in the expanded lattice solves approximatively **CVP** in the original lattice. While this is a heuristic technique, the inexplicably good results of lattice reduction algorithms make it interesting. To the best of our knowledge, the technique was first suggested by Kannan [Kan83] although the technique itself is nowadays considered part of academic folklore.

Suppose we have a basis B and $\mathcal{L} = \mathcal{L}(B)$ and some vector $v \notin \mathcal{L}$ for which we want to solve **CVP**. Then we can try solving it by using a lattice reduction technique on

$$\left[\begin{array}{c|c} 1 & v \\ \hline | & \\ 0 & B \\ | & \end{array}\right], \text{ which “should” output } \left[\begin{array}{c|c} 1 & v' \\ \hline | & \\ 0 & B' \\ | & \end{array}\right]$$

where $\mathcal{L}(B) = \mathcal{L}(B')$ and $\|v\| \geq \|v'\|$ and $(v - v') \in \mathcal{L}$. Therefore the approximate answer to **CVP** would be the vector $w = v - v'$. Now suppose you want to solve **SVP**, but you actually have knowledge on the exact value of some non-zero coefficients, i,e you can construct v such that for a solution s for **SVP** you have $w = s - v$ and $\|w\| < min(\|s\|, \|v\|)$. In that case the transformation is as follows:

$$\left[\begin{array}{c|c} 1 & v \\ \hline | & \\ 0 & B \\ | & \end{array}\right], \text{ which ``should'' output } \left[\begin{array}{c|c} 1 & w \\ \hline | & \\ 0 & B' \\ | & \end{array}\right]$$

and thus you can reconstruct the **SVP** solution $s = v + w$.

2.6 Applications to public-key cryptography

Now that we presented the general problems and algorithms to solve lattice problems we give in this chapter a peek on how those problems are used in cryptography, in particular public-key cryptosystems.

The point of most public-key cryptosystems is to have a public key, which can generate some hard problems, that is available to more parties than the key generator, but that only a few selected parties can make use of the key, i.e a trapdoor, to solve the associated hard problems. In public-key cryptography, we often assume anybody aside from the key owner is a eavesdropper, and no eavesdropper should have any capacity to recover a secret.

We say a cryptosystem is "broken" when a security assumption no longer holds. Of course this definition is very vague: some people would claim that recovering S_k (the secret key, or trapdoor) from P_k (the public key, the "lock") in 45 years instead of an expected 50 years can be considered as "breaking" a system but some others may not see it as a "break", but recovering a secret in 2 months instead of 50 years would be reasonably and unanimously called a "break". There is no proper definition of what it exactly means to "break" a cryptosystem, and we will leave that aside from now on and just describe overall frameworks.

Let us define here P as the set of problems that can be solved in polynomial time with a classical computer, QP the set of problems that can be solved in polynomial time given a *quantum computer*, and NP the set of problems which valid solutions can be checked in polynomial time. Overall, public-key cryptography relies on assumptions we do not know proven:

- Is there *one-way* functions, i.e easy to compute but hard to invert? Is $P \neq NP$?

- If yes to the previous question, do we have *one-way* functions with trapdoors, i.e are *one-way* except in the presence of a constant hint independent of the input?
- If yes to the previous question, do they still exist in the quantum setting? Is $QP \neq NP$?

Any proven *no* to those previous questions basically means the end of public-key cryptography, if one forges a counterexample. For the rest of the thesis, we have to assume those answers are yes. The question is often to determine if the functions we use are actually *one-way* functions with trapdoors. Note that a problem does not have to be NP-hard to be used in cryptography. The most blatant example is the Rivest-Shamir-Adleman (**RSA**) cryptosystem [RSA78]. Its hardness was initially based on the factorization of numbers, and we know it is not a NP-hard problem (unless P=NP): it is actually in a class we call NP-intermediate. Yet, we still use it nowadays[b]: asymptotical complexity aside there is no "practical" attacks to break keys or recover messages as of "right now". It is likely that even if the post-quantum cryptosystems were to be proven not to be NP, they would still be usable as long as no "quasi-polynomial" time quantum algorithm is known to "break" the cryptosystems.

Thus, the goal of post-quantum cryptography is to conceive cryptographic tools that are usable right now, or at least in the very near future, but would remain safely usable in decades if quantum computers become physically available to any entity.

2.6.1 Encryption and Key Encapsulation Mechanism

Public-key encryption schemes are used to encode a message such that the only person who can learn about the content is the sender and the key owner. Here we describe what we call KEM. In a sense, it's like sending a paper mail and ensuring the postman (let us call her Eve) cannot learn anything about the content of the mail even if he wishes to. It is usually made of three primitives functions:

1. **KeyGen**$(s)\rightarrow$ (S_k, P_k) creates a secret key S_k and a public key P_k from a seed s.
2. **Enc**$(P_k,p)\rightarrow$ c uses a key P_k and a plaintext p output a ciphertext c.
3. **Dec**$(S_k,c)\rightarrow$ p uses a key S_k and a ciphertext c output a plaintext p.

[b] whether we should actually use it today is another problem we leave aside for the sake of the example

Supposing (S_k, P_k) were created by the same function call to **KeyGen**(s), the condition **Dec**$(S_k$,**Enc**$(P_k,p))=p$ must always be verified for any valid p. The set of valid p is usually called the "message space". Therefore, if Alice wants to send a secret message p to Bob, the usage of a KEM scheme is as follows:

1. Bob generates a pair of keys (S_k, P_k) from **KeyGen**.
2. Bob sends P_k to Alice, keeps S_k for himself.
   Postwoman Eve can see P_k.
3. Alice sends c to Bob, created using **Enc**(P_k,p) and keeps p for herself.
   Postwoman Eve can see c.
4. Bob recovers p, created using **Dec**(S_k,c).

There are several ways to attack this pattern. What if Alice is malicious and wants to recover S_k? What can Eve learn about the secrets p/S_k if she modifies c/P_k? Answering those questions in the general sense is hard, but ultimately a "secure" KEM should deal with those issues.

2.6.2 Public-Key Digital Signature Schemes

Public-key signature schemes are used to sign a message such that everybody can verify that this signature was produced by a public key knowing the original message. In a sense, it is maybe safer than a classic paper and pen signature: we do not need a previous signature to ascertain the key owner is actually the signatory, and in some cases seeing the signature once is not enough to replicate a valid signature for a different document. It is usually made of three primitives functions:

1. **KeyGen**$(s)\rightarrow$ (S_k, P_k) creates a secret key S_k and a public key P_k from a seed s.
2. **Sign**$(S_k,d)\rightarrow$ s uses a key S_k and a document d to output a signature s.
3. **Verif**$(P_k,d,s)\rightarrow$ **BOOL** uses a key P_k, a document d, a signature s,
   and output **TRUE** if the signature is valid or **FALSE** otherwise.

Supposing (S_k, P_k) were created by the same function call to **KeyGen**(s), the condition **Verif**$(P_k,d$,**Sign**$(S_k,d))=$**True** must always be verified for any valid d.

Therefore, if Bob wants to certify his identity to Alice, the usage of the signature scheme is as follows:

1. Bob generates a pair of keys (S_k, P_k) from **KeyGen**.
2. Bob sends P_k to Alice, keeps S_k for himself.

3. Alice sends a document d for Bob to sign.

4. Bob signs d and sends s to Alice created using **Sign**$(S_k$,$d)$.

5. Alice verifies if **Verif**$(P_k$,d,$s)$ is **TRUE**.

Ensuring a signature scheme is secure follows the same philosophy as described in the previous sections.

2.6.3 Knapsack cryptosystems

We earlier mentioned how knapsack cryptosystems are related to lattices. Let us give an example of a knapsack system. Typically, knapsack cryptosystems are used for encryption and not for signatures and used to be very popular for their simplicity both in concept and in computational complexity. A short introduction can be found in [Odl90].

Typically, given a "dimension" n, a classical additive knapsack cryptosystem would be based around the 0-1 knapsack and look like the following:

1. **KeyGen()**$\rightarrow (T, k)$ creates a trapdoor T and a public vector $k \in \mathbb{N}^n$.

2. **Enc**$(k$,$s)$$\rightarrow e$ output $e = \langle k, s \rangle$ where $s \in \{0, 1\}^n$

3. **Dec**$(T$,$e)$$\rightarrow p$ solves the knapsack problem on the set s and target e and recover s.

The most famous example of knapsack cryptosystems is the famously broken Merkle-Hellman cryptosystem [MH78]. See Alg 10 for **KeyGen()** and Alg 11 for **Dec()**. We assume **Enc()** is just an inner product and does not need a description.

Algorithm 10 Merkle-Hellman Key Generation

```
Require: A dimension $n$, and initialized random number generators
Ensure: A trapdoor $T$ and its related public vector key $k$
Randomly pick $v \leftarrow (v_1, ..., v_n)$ such that $v_i > \sum_{j=1}^{i-1} v_j$    ▷ Superincreasing sequence
Randomly pick $M > \|v\|_1$
Randomly pick $R$ such that $\mathrm{GCD}(R, M) = 1$    ▷ Pick invertible modulo $M$
$T \leftarrow (v, M, R)$
$k \leftarrow R^{-1} \times v \mod M$    ▷ Multiply $v$ by modular inverse
return $(T, k)$
```

The Merkle-Hellmann cryptosystem is quite simple to implement and its high popularity was also due to its small arithmetical cost: linear in both the size of the moduli and the dimension. However it have been broken in several ways [Bri83, Sha82]

Algorithm 11 Merkle-Hellman Key Decryption

```
Require: A trapdoor $T = (v, M, R)$ and an integer $e$.
Ensure: A vector $s$ such that $R^{-1} \times \langle v, s \rangle \mod M = e$    ▷ i.e Enc($k$,$s$)= $e$
$e \leftarrow R \times e \mod M$    ▷ Transform $\langle k, s \rangle$ to $\langle v, s \rangle$
$s \leftarrow (0, \ldots, 0)$
for $i = n$ down to 1 do
    if $e \leq v_i$ then $s_i = 1$, $e \leftarrow e - v_i$    ▷ Make use of $v_i > \sum_{j=1}^{i-1} s_j v_j$
return $s$
```

Solve **SVP** on $\mathcal{L} = \mathcal{L}(B)$ such that

$$B = \begin{bmatrix} 235 & 0 & 0 & 0 & 0 & 0 \\ 78 & 1 & 0 & 0 & 0 & 0 \\ 54 & 0 & 1 & 0 & 0 & 0 \\ 65 & 0 & 0 & 1 & 0 & 0 \\ 85 & 0 & 0 & 0 & 1 & 0 \\ 92 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Consider $S = \{1, 78, 54, 65, 85, 92\}$

Let $v \in \{0, 1\}^n$, such that $235 = \sum_{i=1}^{n} v_i S_i$.

Find v, i.e solve the **0-1 KP**.

Both answers are $v = [0, 1, 0, 1, 0, 1]$.

Figure 2.13: Lattice on left, knapsack on right

and even some extensions [VAN94] have been broken by lattice reduction techniques [NS97]. Even multiplicative knapsacks (using a "non-additive" group [CR88]) have been broken also with lattice reduction techniques [SH95, Vau98]. As a side note, it seems like even outside the field of cryptography knowing where the hardness of knapsacks lie was problematic [Pis05]. It seems like density of the knapsack was one of the keys to understand the hardness of knapsacks: if density was low then it could be broken easily [LO85, CJL+92], and if it was close to 1 then the problem is harder to tackle on [HGJ10, BCJ11]. Betting on high-density to create secure knapsack cryptosystems was considered [WWH07, WH10] but often quickly shot down [You09, Lee11].

Without going to the specifics, we show a graphical example on how those two relates in figure 2.13. While it is a bit of an exaggeration, further graphical representations can be found in [SE94, PSZ12].

Historically, lattice-based cryptography is actually the continuation of knapsack-based cryptography. The instances are just slightly different: we consider the modular knapsack and the message spaces are extended to small negative integers rather than just positive ones. Some papers in lattice-based cryptography actually describe the construction of their lattices as knapsack instances [Mic07], and the other way around also happens: any lattice-based scheme using a tiny set s as the base of a message space s^n is nothing more than a particular knapsack.

2.6.4 Specific lattices in crypto

Cryptography set aside, if we wish to study random lattices it would make more sense to rely on the distribution given by Goldstein and Mayer [GM03], which mostly gives perfect **HNF** (even though experimentally this seems true only after permutation, but that would be a point of another chapter). However in cryptography we aim to have low data sizes for our instances, an extremely costly computational problem for an attacker to solve (i.e unfeasible) and a trapdoor to make the problem computationally easy (i.e instantaneous) for the person who generated the instances. This have led to various studies on what makes a lattice problem hard and what types of lattice can we use for both security and efficiency. The most famous structured lattices are described in the following, and we only give the necessary background for the work we present (in particular, chapter 6).

Definition 34 (q-ary lattices)**.**
We call a lattice $\mathcal{L} \subset \mathbb{Z}^n$ a modular lattice if there exists an integer $q < vol(\mathcal{L})$ such that $q\mathbb{Z}^n \subseteq \mathcal{L}$. To precise the value q we say $\mathcal{L}$ is a q-ary lattice.

The precision $q < \det(\mathcal{L})$ is important since otherwise every full-rank lattice is a modular lattice as $\det(\mathcal{L})\mathbb{Z}^n \subset \mathcal{L}$. To convince yourself, take any vector of $\mathbb{Z}^n$, multiply it by $\det(\mathcal{L})$ and apply the reduction from the example 2. As a matter of fact, if both the top-upper part and the bottom-right part of the **HNF** of $\mathcal{L}$ is diagonal and every diagonal coefficient divides q then $\mathcal{L}$ is a q-ary lattice. The reverse is not necessarily true: given a **HNF** H, if for all i, $H_{i,i} = d$ and $d_i \mid q$, then for example $H_{i,j} = kd < H_{j,j}$ for $k \in \mathbb{Z}$ and $j < i$ is sufficient.

Example 3. *Examples of* **HNF** *basis of q-ary lattices when $q = 26$:*

$$\begin{bmatrix} 13 & 0 & 0 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 & 0 & 0 \\ 0 & 0 & 13 & 0 & 0 & 0 \\ 12 & 4 & 8 & 1 & 0 & 0 \\ 3 & 7 & 6 & 0 & 1 & 0 \\ 11 & 9 & 10 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 26 & 0 & 0 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 19 & 4 & 1 & 1 & 0 & 0 \\ 8 & 7 & 1 & 0 & 1 & 0 \\ 4 & 5 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 26 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 19 & 4 & 0 & 1 & 0 & 0 \\ 9 & 17 & 0 & 0 & 1 & 0 \\ 2 & 9 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 26 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 0 & 0 & 0 & 0 \\ 4 & 12 & 2 & 0 & 0 & 0 \\ 19 & 14 & 0 & 1 & 0 & 0 \\ 23 & 7 & 0 & 0 & 1 & 0 \\ 21 & 19 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 13 & 0 & 0 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 & 0 & 0 \\ 8 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13 & 0 & 0 \\ 3 & 7 & 4 & 2 & 1 & 0 \\ 2 & 9 & 3 & 12 & 0 & 1 \end{bmatrix}$$

In cryptography however, there is usually only one **HNF** representation of q-ary lattices people consider for dimension $n > m > 1$: upper-right is $q \times Id_m$, bottom-left

Id_{n-m}. Of course, that is assuming we consider row vectors and the **HNF** definition is set as bottom triangular. Those lattices hold the property that their basis do not change by the following transformation $\mathcal{L} \leftarrow \mathcal{L} \cup q\mathbb{Z}^n$, which led us to often consider computations over $\mathbb{Z}_q$ rather than $\mathbb{Z}$.

Definition 35 (Principal Ideal lattices).
Let $R_f = \mathbb{Z}[X]/\langle f \rangle$ a ring of polynomials over integers where f is a monic polynomial of degree n, and consider the map $\Phi_f : X^{(i-1)} \rightarrow v_i$ where v_i is the i−th vector of the identity basis in $\mathbb{Z}^n$ and X^i the monomial of degree i over R_f. We can then define a non-trivial ring structure over a particular set of sublattices of $\mathbb{Z}^n$ by mapping R_f to $\mathbb{Z}^n$ via Φ. Suppose we take an ideal of R_f, namely $R_f(g) = \langle g \rangle$ with $g \in R_f$. We denote $\mathcal{L}(g, f)$ the lattice $Im_{\Phi_f}(R_f(g))$.
For any lattice $\mathcal{L}$, if there exists g, f such that $\mathcal{L} = \mathcal{L}(g, f)$, we say $\mathcal{L}$ is a principal ideal lattice.

Note that if g is replaced by a set of polynomials $< g_1, ..., g_k >$ to generate an ideal lattice, then it is unclear if the resulting lattice is still a *principal* ideal (i.e the existence of a single tuple g, f), but is an ideal lattice nevertheless[c]. For the rest of this thesis however, whether an ideal lattice is principal or not does not have much of an impact. After all, none of our published works are specifically about ideal lattices but could eventually apply to them after further studies.

Ideal lattices have another structure where efficiency is greatly improved, although there is no improvement on the security (it is up to debate if the security is actually downgraded). The morphism Φ is applied as the following:

$$\Phi(\sum_{i=0}^{n-1} a_i x^i) = [a_0, a_1, ..., a_{n-1}].$$

Suppose now the we are given an ideal lattice $\mathcal{L}$ of determinant P and admits a perfect **HNF** as a basis, thus ensuring $[P, 0, ..., 0] \in \mathcal{L}$. Then if we know another vector $v = [-\gamma, 1, 0, ..., 0] \in \mathcal{L}$, then we can write a basis in the form

$$\begin{array}{rcl} \begin{matrix} P \\ v \\ v \times X \\ v \times X^2 \\ v \times X^3 \\ v \times X^4 \end{matrix} & \begin{matrix} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{matrix} & \begin{bmatrix} P & 0 & 0 & 0 & 0 & 0 \\ -\gamma & 1 & 0 & 0 & 0 & 0 \\ 0 & -\gamma & 1 & 0 & 0 & 0 \\ 0 & 0 & -\gamma & 1 & 0 & 0 \\ 0 & 0 & 0 & -\gamma & 1 & 0 \\ 0 & 0 & 0 & 0 & -\gamma & 1 \end{bmatrix} \end{array}, \text{ reduces to } \begin{bmatrix} P & 0 & 0 & 0 & 0 & 0 \\ -\gamma & 1 & 0 & 0 & 0 & 0 \\ -\gamma^2 & 0 & 1 & 0 & 0 & 0 \\ -\gamma^3 & 0 & 0 & 1 & 0 & 0 \\ -\gamma^4 & 0 & 0 & 0 & 1 & 0 \\ -\gamma^5 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

[c] Determining whether an ideal is principal or not does not seem to be easy for any general ideal: it is one of the main problems of algebraic number theory [Coh93]. It is nevertheless doable and MAGMA provide a function for that: `http://magma.maths.usyd.edu.au/magma/handbook/text/394#4026`. It is probably simpler for the ideals used in cryptography when using norm arguments.

i.e we can get a **HNF** for almost free where $(-\gamma, P)$ are enough to define the whole lattice. Note that given such a form, we can always extract an ideal lattice: let $g = X - \gamma$ and $f = X^n - \gamma X^{n-1} - P$, then $\mathcal{L}(g, f)$ generates the lattice we described. However such forms are currently not used in cryptography.

Property 3. *Let $\mathcal{L}$ be a lattice such that $\mathcal{L} = \mathcal{L}(g, f)$. There exists an infinite amount of tuples (g', f') such that $\mathcal{L} = \mathcal{L}(g', f')$.*

In cryptography they tend to use cyclic lattices ($f = X^n - 1$) with prime n or anti-cyclic ($f = X^n + 1$ first used in [Mic07]) where n is a power of two. Cyclotomic polynomials are used and discussed in [LPR10]. While changing the quotient have no influence on the compression method we just showcased, quotients are either kept for security assumptions (lack of a "weak" Chinese Remainder Theorem (CRT) decomposition) or for faster ring-specific operations (usage of Number Theoretic Transform (NTT) for example). While it is unclear if the "ring" version of all the problems we defined in section 2.4 are actually hard, it reminds an open problem. The density of such lattices that are chosen for cryptography is exponentially vanishing unlike co-cyclic lattices [BL09]. So far, it does not seem to be a worry for the security. To compromise between efficiency and potential security, a "module" version that is less compact that the "ring" version has been proposed [BGV14]: in a sense, instead of one single large "ring" basis, we concatenate multiple smaller "ring" basis to create one large basis of a new lattice. This multi-block structure is assumed to be safer than the single-block version. The block decomposition approach can probably be applied in other ways than in the ideal case: as far as we know we have not seen any proposition (aside maybe with the tensorial product [FS99]).

What we present now is one of the oldest structured lattices, if not the oldest, that we found in the cryptographic academic literature:

Definition 36 (Diagonal Dominant Lattices)**.**
We say a lattice is a diagonally dominant type lattice (of dimension n) if it admits a diagonal dominant matrix as a basis B as in [BR91], i.e.,

$$\forall i \in [1, n], B_{i,i} \geq \sum_{j=1, i \neq j}^{n} |B_{i,j}|$$

We can also see a diagonally dominant matrix B as a sum $B = D + R$ where D is diagonal and $D_{i,i} > \|R_i\|_1$. To avoid conflicting notations between the diagonal matrix and the diagonal coefficient, we will denote from now on D_{Id} the product of the integer D by the canonical basis Id. We might also denote D_g a diagonal matrix which diagonal coefficients might not all be equals.

However sometimes we need a definition for lattices with a stronger weight on the diagonal while not being diagonal dominant in the mathematical sense:

Definition 37 ("Weakly" Diagonal Dominant Lattices).
We say a lattice is a "weakly diagonally dominant lattice" of dimension n *if it admits a basis of the form* $D_{Id} + N$ *where* $D \in \mathbb{Z}$, *and* $N \in [-\mu, \mu]^{n \times n}$ *a noise matrix.* μ *is called the bound of the noise and* D *is considered much "larger" than* μ.

This basis structure have vectors that are closely orthogonal: the difference between a diagonal dominant basis and its **GSO** is "geometrically low". Therefore it can guarantee long **GSO** vectors that have roughly the same norm and provides a high-decryption capacity via Babai's reduction algorithms. Unfortunately, the computation of the **HNF** is expensive, and key sizes are not trivially compressible (unless combined with the previously mentioned structures).

2.6.5 Other structures and our research direction

Those structures led to cryptographic constructions where the security is based on lattices while not being a pure lattice problem, like Number Theory is Really Useful (**NTRU**), Learning With Errors (**LWE**) and their variants which we will not define in this thesis as we do not make use of them. As far as we know HIMMO also fall under that category [GMRS+15, GMGPG+14], and we do not at the moment have an exhaustive list of all existing structures.

Module and ring structures are currently the most popular structure used in lattice-based cryptography, and its study is still going on actively [LPMSW19]. While most lattice-based submissions accepted on the round 2 of NIST PQC standardization process rely on a module or ideal structure, its recency does not give a lot confidence. To the best of our knowledge, no cryptosystem are proven to be NP-hard, which is not a claim of insecurity by itself but should encourage innovations in the constructions of new mathematical primitives even without any "security proofs". Leaving "security proofs" aside, it does seem that finding generators on some ideal lattices we often use in cryptography[d], i.e the cyclotomic case, is not a hard problem in the quantum case [CDPR16].

The problem is still the minimization of the generator, however as nobody currently has access to a quantum computer, it is unclear if we are facing a case where

[d] which admit a trapdoor, so usually a short generator vector. In the general case, it is possible that the shortest vector is not a generator.

there is a noticeable gap between the upper bound estimated for an algorithm output, and its average output. Experiments have been done with classical computers on more general cases [LPS19, BBdV$^+$17], but those might not be sufficient and the computational effort required in this particular topic seems to be a hindrance for further work. On the other hand, theoretical estimations results are flourishing [PMHS19]. History could show a repeat of **LLL** and **BKZ** where the experimental results behave much better in practice than in theory [GN08]. To learn more about the current state-of-the-art knowledge about the ideal structure, a good entry point would be the survey from Ducas [Duc17].

The rest of this thesis will mostly be focused on structures which are neither ring-based or q-ary based, whether for efficiency or security[e]. We want to stress we do not claim the work we present produces more secure cryptosystems compared to the popular tools of lattice-based crypto: a longer and deeper study from external contributors is at least required to give a closer level of confidence.

Nevertheless, we believe exploring unpopular research avenues should be welcomed and our work is a start, if it ever has to lead somewhere. Also, it could provide some basis for alternative backups if other structure breaks[f].

[e] The last chapter is an outlier.

[f] Let's be honest I am doing this for fun

Chapter 3

Hiding Lattices within Sublattices

This chapter is a rewritten version of the paper published in the Journal of Mathematical Cryptology [SPS19a], which we originally submitted in 2016 first in ASIACRYPT and then to DCC. While the published paper essentially did not change much since the initial version submitted to ASIACRYPT, the successive reviews we received made it clear our writing was unclear. Thus, the redaction of this chapter is very different from the initial paper, and we hope the reader will not have difficulties understanding its content.

The aim of this paper was to present a technique to hide the structural weakness of a lattice, with the example of the Goldreich-Goldwasser-Halevi scheme (**GGH**) as a direct application. We do so with the help of an intersection between the weak lattice used for decryption and a random lattice. This method gives us a new lattice we can provide as a public key, different from the secret key.

But are intersecting lattices a secure structure? While it seems hard to answer the question, an intuition is needed to understand the problem. Therefore, before we describe the patch, we will give some reminders about lattice intersection properties: intersections of lattices in the literature are scarce, at least in cryptography. To the best of our knowledge the few references includes a theorem in Micciancio and Goldwasser's book [MG12], and one paper about a broadcast attack [PS09]. Please note, that every proof that is given might not be new, or might be documented elsewhere in an improved version. However, as we could not find any prior reference, I made proofs myself whenever I could not find previous documentation.

Experts on the field and people who work with Gröbner Basis (seeing intersections of ideals) might want to skip most of the next two sections: while we believe this was previously undocumented, except of course in the work we published, expert readers might intuitively know most of the properties with another point of view.

Let us first introduce some formulas from [MG12]:

Property 4 (Determinant of sub-lattices)**.**
Let $\mathcal{L}_1, \mathcal{L}_2$ *be two lattices.* $\mathcal{L}_1 \subset \mathcal{L}_2$*, then* $\det(\mathcal{L}_2) \mid \det(\mathcal{L}_1)$.

From this property, basic arithmetic shows that for any *integral* lattices $\mathcal{L}_1, \mathcal{L}_2$ we have $\det(\mathcal{L}_1) \leq \det(\mathcal{L}_1 \cap \mathcal{L}_2) \leq \det(\mathcal{L}_1) \times \det(\mathcal{L}_2)$.

Property 5 (Lattice intersection (for full-rank lattices))**.**
Let $\mathcal{L}_1, \mathcal{L}_2$ *be two full-rank lattices. Then* $\mathcal{L}_1 \cap \mathcal{L}_2 = (\mathcal{L}_1^\times \cup \mathcal{L}_2^\times)^\times$.

The computation of the dual, while doable in polynomial time, can be expensive for practical applications. Which is why we explicit here a way to compute intersections given the Hermite Normal Form (**HNF** see definition 9 in background) of lattices. Computing a **HNF** might be worse than computing duals if you are given a reduced basis, however in cryptographic applications a **HNF** is preferred as a public key. The other motivation is the fact that generation of random **HNF** with a chosen determinant is trivial (following the distribution of [GM03]). This gives enough motivation to see intersections from **HNF**.

3.1 A HNF vision of lattice intersections

Supposing we are working on a full-rank square Hermite Normal Form (**HNF**) matrix $M \in \mathbb{N}^{n \times n}$. To consider all vectors of $\mathcal{L}(M)$ of the form $(v_1, ..., v_{i-1}, v_i, 0, ..., 0) \in \mathbb{Z}^n$, is to consider instead of $< M_i >_{\mathbb{Z}}$ the i-dimensional lattice $< M_1, ..., M_i >_{\mathbb{Z}}$. Those trivial facts are going to provide the necessary arguments for all following properties. The following property uses the uniqueness of the **HNF** for vectors $(v_1, ..., v_{i-1}, v_i \neq 0, 0, ..., 0) \in \mathbb{Z}^n$.

Property 6 (Unique positive coefficients property)**.**
Let $M \in \mathbb{N}^{n \times n}$ *be a full-rank square* ***HNF*** *matrix and* $i \in [1, n]$. M_i *represents the unique vector with the smallest coefficients* $0 < M_{j,i} < M_{j,j}$ *for* $j < i$ *and* $M_{i,i} > 0$ *such that* $< M_i >_{\mathbb{Z}}$ *is an one-dimensional lattice containing vectors of* $\mathcal{L}(M)$ *of the form* $(v_1, ..., v_{i-1}, v_i \neq 0, 0, ..., 0) \in \mathbb{Z}^n$ *(and the all-zero vector). Therefore, if* $v \in \mathcal{L}(M)$ *and* $v = (v_1, ..., v_{i-1}, v_i \neq 0, 0, ..., 0)$*, then* $M_{i,i} \mid v_i$.

Proof. To prove the divisibility by the head coefficient, we can make a proof by absurd. Suppose there is another vector $v \in \mathcal{L}(M)$ that is of the form $(v_1, ..., v_{i-1}, v_i > 0, 0, ..., 0) \in \mathbb{Z}^n$, such that $v_{i,i} < M_{i,i}$. Then the Gauss-Jordan reduction with the

HNF cannot reduce $v_{i,i}$ to 0, thus $v \notin \mathcal{L}(M)$, which is absurd. Now for the unicity. If there is another different vector $a \in \mathcal{L}(M_1, ..., M_i)$ such that $a_i = M_{i,i}$ and $0 < a_j < M_{j,j}$ for $j < i$, then for some j there must be $0 < |a_j \pm M_{i,j}| < M_{j,j}$, thus not reducible to 0 via $< M_1, ..., M_i >_{\mathbb{Z}}$. However both of them are supposed to be in $< M_1, ..., M_i >_{\mathbb{Z}}$, which is a group, thus this is absurd. □

This is due to the uniqueness of the **HNF** and basic linear algebra, and we have not yet talked about intersections, which we will do now. The first property on intersections we are going to present was not on the original paper: however as we had comments on the matter during oral presentations (special thanks to Pr Jean-Claude Bajard), we decided therefore to mention the property in this thesis. An existence of the **HNF** of the intersection is always guaranteed. What is less obvious is the guarantee that the intersection of two integer lattices with square and full-rank basis also have a basis that is full-rank. After all, the intersection of two discrete sets do not need to exist: it can be empty. In the lattice case, the intersection is at least the all-zero vector. However in does not have to be full-rank for the non-integral case: $\mathbb{Z}^n \cap (\sqrt{2}\mathbb{Z}^n) =< 0 >_{\mathbb{Z}}$. And even in the integral case, the intersection can be $< 0 >_{\mathbb{Z}}$ if they are not full rank, like the direct sum case: $< (0, 1) >_{\mathbb{Z}} \cap < (1, 0) >_{\mathbb{Z}} =< 0 >_{\mathbb{Z}}$. Full-rank integral lattices however, are a particular case.

Property 7 (Full-rank of the intersection of full-rank integer lattices).
Let $A, B \in \mathbb{Z}^{n \times n}$ be full rank. Then $\dim(\mathcal{L}(A) \cap \mathcal{L}(B)) = n$, thus the intersection admits a full-rank basis.

Proof. We just have to prove that there exists a full-rank sublattice. In that regard, we want to show it admits a triangular basis. The simplest lattice can come to mind. Since A, B are non-singular integral matrices, $\det(A), \det(B)$ are non-zero integers. Therefore consider the matrix $C = \det(A) \times \det(B) \times Id_n$. C has trivially full-rank, and for all $i \in [1, n]$, $C_i \in \mathcal{L}(A) \cap \mathcal{L}(B)$. $\mathcal{L}(C)$ is then full-rank and part of $\mathcal{L}(A) \cap \mathcal{L}(B)$, which is sufficient. □

Now that we know its existence, we need to find methods to compute them efficiently. To warm up, we start first with the most trivial property:

Property 8 (Conservation of Row Linearity).
*Let $A, B \in \mathbb{Z}^{n \times n}$ in **HNF** form. If for some $i \in [1, n]$ and $\alpha \in \mathbb{N}^*$, $\alpha A_i = B_i$ then the **HNF** basis C of $\mathcal{L}(A) \cap \mathcal{L}(B)$ verifies $C_i = B_i$.*

Proof. If $< A_i >_{\mathbb{Z}} \supseteq < B_i >_{\mathbb{Z}}$, then their intersection is $< B_i >_{\mathbb{Z}}$. Therefore the unique one-dimensional lattice we pointed out previously $< C_i >_{\mathbb{Z}}$ for $\mathcal{L}(A) \cap \mathcal{L}(B)$ must also be $< B_i >_{\mathbb{Z}}$. □

Every computation of two **HNF** intersection can then be simplified by removing each common i-th row when their other non-diagonal coefficient are 0 and the resulting zero columns, and reinserting them later. This always apply when the diagonal coefficient is 1 and there is zero values below, however it is unlikely that two random lattices of non-small determinants would share similar rows and have zeroes above. A natural question would be why do we not make a generalization for $C_i = \alpha A_i + \beta B_i$: this is because it is not true for $i > 1$. This is trivially true thanks to the "cascading" property that a **HNF** gives: for a matrix in **HNF** form A and $i < j < k$, changes on A_i can have repercussions on A_j if A is to be kept a **HNF**, but modifications A_k will have no effect on A_i. This also trivially influences the intersection:

Property 9 ("Cascading" influence on the intersection).
Let $A, A', B, B' \in \mathbb{Z}^{n\times n}$, all non-singular in lower triangular form. If for some $i \in [1, n]$,

$$< A_1, ..., A_i >_{\mathbb{Z}} = < A'_1, ..., A'_i >_{\mathbb{Z}} \textit{ and } < B_1, ..., B_i >_{\mathbb{Z}} = < B'_1, ..., B'_i >_{\mathbb{Z}}$$

*then for C the **HNF** basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$ and C' the **HNF** basis of $\mathcal{L}(A') \cap \mathcal{L}(B')$,*

$$\forall j \in [1, i],\ C_i = C'_i$$

We do not think a proof is needed, but it is a property that must be kept in mind to make further proofs easier.

However, we can deduce other properties. The following is another consequence of the previous observations:

Property 10 (Row coefficient head divisibility).
*Let $i \in [1, n]$ and $A, B, C \in \mathbb{Z}^{n\times n}$ are full-rank in **HNF** such that $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$.*

$$\textit{Then } \mathrm{lcm}(A_{i,i}, B_{i,i}) \mid C_{i,i} \mid \mathrm{lcm}(\textstyle\prod_{k=1}^{i} A_{k,k}, \prod_{k=1} B_{k,k}))$$

Proof. The left part is easy and follows from the previous properties. The right part follows from $\det(< A_1, .., A_i >_{\mathbb{Z}}) = \prod_{k=1}^{i} A_{k,k}$, $\det(< B_1, .., B_i >_{\mathbb{Z}}) = \prod_{k=1}^{i} B_{k,k}$ as they are triangular. Let v be the i-th row of $\mathrm{lcm}(\prod_{k=1}^{i} A_{k,k}, \prod_{k=1} B_{k,k}) \times Id_n$. $v \in \mathcal{L}(A) \cap \mathcal{L}(B)$, thus $C_{i,i}$ must divide $v_i = \mathrm{lcm}(\prod_{k=1}^{i} A_{k,k}, \prod_{k=1} B_{k,k})$. □

Note like unlike the previous case, this property does not allow to remove rows.

Then another special case, by specialized application of the previous property:

Property 11 (Diagonal of Intersection of Successive Coprime Diagonals).
*Let $A, B \in \mathbb{Z}^{n\times n}$ be two full-rank integer matrices in **HNF**, such that we have for $k \leq n$*

$$\forall i \in [1,k],\ \text{GCD}(A_{i,i}, B_{i,i}) = 1$$

*Then $\mathcal{L}(A) \cap \mathcal{L}(B)$ admits as a **HNF** basis the matrix C such that*

$$\forall i \in [1,k],\ C_{i,i} = A_{i,i} \times B_{i,i}$$

Proof. $A_{i,i} \times B_{i,i} \mid C_{i,i}$ by the previous property. What remains to be proven is the other way around. By successive iterations from $i = 1$ to $i = k$ for $C_{i,i}$ over the lattice $< A_1, .., A_i >_{\mathbb{Z}} \cap < B_1, .., B_i >_{\mathbb{Z}}$, one can see that applying the previous inequality and the fact that for any lattices $\mathcal{L}_1, \mathcal{L}_2$, $\det(\mathcal{L}_1 \cap \mathcal{L}_2) \leq \det(\mathcal{L}_1) \times \det(\mathcal{L}_2)$ i.e $\prod_{k=1}^{i} C_{k,k} \leq (\prod_{k=1}^{i} A_{k,k}) \times (\prod_{k=1}^{i} B_{k,k})$, $C_{i,i}$'s has only one possibility. □

To simplify notations, we denote $A \cap B = C$ as "C is a (**HNF**) basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$".

Example 4. *A, B, C three **HNF** and $A \cap B = C$*

$$A : \begin{bmatrix} A_{1,1} & 0 & 0 & 0 & 0 \\ A_{2,1} & 1 & 0 & 0 & 0 \\ A_{3,1} & 0 & A_{3,3} & 0 & 0 \\ A_{4,1} & 0 & A_{4,3} & 1 & 0 \\ A_{5,1} & 0 & A_{5,3} & 0 & 1 \end{bmatrix}, B : \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & B_{2,2} & 0 & 0 & 0 \\ 0 & B_{3,2} & 1 & 0 & 0 \\ 0 & B_{4,2} & 0 & B_{4,4} & 0 \\ 0 & B_{5,2} & 0 & B_{5,4} & 1 \end{bmatrix}, C : \begin{bmatrix} A_{1,1} & 0 & 0 & 0 & 0 \\ - & B_{2,2} & 0 & 0 & 0 \\ - & - & A_{3,3} & 0 & 0 \\ - & - & - & B_{4,4} & 0 \\ - & - & - & B_{5,4} & 1 \end{bmatrix}$$

We stress that this property is only true when the *successive first* diagonal elements are actually coprime. If there is one diagonal coefficient in the middle which does not respect coprimality, the GCD factor "moves down" in most cases. We will explain this phenomena later.

For now, we will first focus on the case of perfect **HNF**. The following is, albeit trivial, the building block of our intersection computations and a direct result of our previous property:

Property 12 (CRT decomposition of perfect **HNF**).
*Let $A, B \in \mathbb{Z}^{n \times n}$ be full-rank in perfect **HNF**. If* $\text{GCD}(A_{1,1}, B_{1,1}) = 1$*, then*

$$C_{1,1} = A_{1,1} \times B_{1,1}$$
$$\forall i \in [2,n],\ C_{i,1} = A_{i,1} \mod A_{1,1}$$
$$\forall i \in [2,n],\ C_{i,1} = B_{i,1} \mod B_{1,1}$$

*where C is the **HNF** basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$.*

Proof. There is an unique solution for the diagonal coefficients by the previous property, and the solution for the coefficients below the diagonal can be trivially checked to be unique. □

Example 5. *Intersections of perfect **HNF** of co-prime determinants*

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 \\ 19 & 1 & 0 & 0 & 0 \\ 11 & 0 & 1 & 0 & 0 \\ 13 & 0 & 0 & 1 & 0 \\ 17 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This property basically states that the intersection of two perfect **HNF** basis of coprime determinants is then computed from 1 integer product and $n-1$ CRT recompositions from the same ring. The other consequence of this property is also to consider that any perfect **HNF** can be decomposed into a product of “prime” overlattice, i.e overlattice of same rank with prime power determinant. From now on, we can generalize all next results on diagonal coefficients being prime numbers p by looking at a CRT decomposition of the elements below the diagonal (modulo the prime powers factors of the diagonal elements). Now that we know what happens with coprime determinants, the natural question is to determine what happens when two determinants are prime and equals considering perfect **HNF**. This is where the previous mention about the gcd factor “moving down” makes sense. Which gives us the following property:

Property 13 (GCD descent in the intersection)**.**
*Let p be a prime number, $A, B \in \mathbb{Z}^{n\times n}$ are full-rank and in **HNF**, such that* $\det(A) = p$ *and* $\det(B) = p^k$ *for* $0 < k < n$*, and C the **HNF** basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$. If $A_{1,1} = p$ and $< B_1, ..., B_k >_{\mathbb{Z}} = \mathcal{L}([p \times Id_k \mid 0])$, then $< C_1, ..., C_k >_{\mathbb{Z}} = \mathcal{L}([p \times Id_k \mid 0])$, and*

If $B_{k+1} \notin \mathcal{L}(A)$, then $< C_1, ..., C_{k+1} >_{\mathbb{Z}} = \mathcal{L}([p \times Id_{k+1} \mid 0])$.
If $B_{k+1} \in \mathcal{L}(A)$, then $C_{k+1} = B_{k+1}$.

Proof. $< C_1, ..., C_k >_{\mathbb{Z}} = \mathcal{L}([p \times Id_k \mid 0])$ is easy to see as $< A_1, ..., A_k >_{\mathbb{Z}} \supseteq \mathcal{L}([p \times Id_k \mid 0])$.
Any C_{k+1} represents the unique solution $C_{k+1} = \alpha_1 A_1 + ... + \alpha_{k+1} A_{k+1} = \beta_1 B_1 + ... + \beta_{k+1} B_{k+1}$, where $\beta_{k+1}, \alpha_{k+1}$ are both minimized and strictly positive.
Since $< B_1, ..., B_k >_{\mathbb{Z}} \subset < A_1, ..., A_k >_{\mathbb{Z}}$, we now have $C_{k+1} = \alpha_1 A_1 + ... + \alpha_{k+1} A_{k+1} = \beta B_{k+1}$.
“If $B_{k+1} \in \mathcal{L}(A)$, then $C_{k+1} = B_{k+1}$.” is thus trivial by setting $\beta = 1$.
The other implication is left to prove when $B_{k+1} \notin \mathcal{L}(A)$.
$\det(A) = p$, $A_{1,1} = p$, $\det(B) = p^k$ and $\prod_{i=1}^{k} B_{i,i} = p^k$ implies $A_{k+1,k+1} = B_{k+1,k+1} = 1$. Thus $C_{k+1,k+1} = \beta = \alpha_{k+1}$ from the **HNF**’s triangular form, and must divide p by a previous property, therefore is either 1 or p.
$C_{k+1,k+1} = 1$ implies $\alpha_{k+1} = \beta_{k+1} = 1$ but contradicts $B_{k+1} \notin \mathcal{L}(A)$. Thus $C_{k+1,k+1} = p$. From there we know every other non-zero coefficient of pB_{k+1} is reducible to the zero vector in $< C_1, ..., C_k >_{\mathbb{Z}}$, giving finally $< C_1, ..., C_{k+1} >_{\mathbb{Z}} = \mathcal{L}([p \times Id_{k+1} \mid 0])$. □

This property basically explains that intersecting randomly k random lattices admitting perfect **HNF** of prime determinant q as basis will give something eerily familiar to a q-ary lattice with very high probability. The GCD q "flows down" the diagonal, "cleaning" the non-zero coefficients on each row. For $q >> n$, n intersections of random lattices of determinant q will highly likely give $q \times Id_n$ as a result.

Example 6. *Successive intersections. Note how we only change a row each time. Changing multiple rows lead to the same phenomena, except for the bottom columns having different coefficients below the diagonal.*

$$\textit{2 lattices:} \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\textit{3 lattices:} \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\textit{4 lattices:} \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\textit{5 lattices:} \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Note that what the proof essentially gives is the multiplication by p of the diagonal coefficient. It can therefore be generalized to transform cases where a diagonal term p transforms into p^2 when the coefficients on the same row were non-zero. However for the rest of the work, we do not need such generalizations. Now the question is the following: is it hard to actually compute such intersections when more than one successive row differ? Since the point of this section was to show the computational advantages of strictly using the **HNF** when available for free, we will now give an example of formulas for fast computation:

Property 14 (Intersection of "equal prime determinants" perfect **HNF**).
*Let $A, B \in \mathbb{Z}^{n\times n}$ be full-rank perfect **HNF**, with no equal rows except for $A_{1,1} = B_{1,1}$. Let $A_{1,1}, B_{1,1}$ be a prime number p. Then* $\det(\mathcal{L}(A) \cap \mathcal{L}(B)) = p^2$ *and C its **HNF** is*

$$\begin{gathered} C_{1,1} = C_{2,2} = p,\ C_{2,1} = 0 \\ \forall i \in [3, n],\ C_{i,2} = (B_{i,1} - A_{i,1}) \times (A_{2,1} - B_{2,1})^{-1} \textit{ mod } p \\ \forall i \in [3, n],\ C_{i,1} = C_{i,2}A_{2,1} + A_{i,1} = C_{i,2}B_{2,1} + B_{i,1} \textit{ mod } p \end{gathered}$$

Proof. The first two vectors result from the GCD descent property. Now we only have to determine all the rest. From the "Cascading" influence we know we can generalize a formula on a row to all others, as they don't influence each other and share the same non-zero coefficient (namely 3) and the same relationship with the first two vectors. Given $i > 3$, we must have some linear combination:

$$C_i = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_i A_i = \beta_1 B_1 + \beta_2 B_2 + \beta_i B_i$$

Note that since the upper part is $p \times Id_2$, we can essentially work modulo p, thus giving

$$C_i = \alpha_2 A_2 + \alpha_i A_i = \beta_2 B_2 + \beta_i B_i \text{ mod } p$$

From the previous property $C_{i,i} = 1$, we can already set $\alpha_i = \beta_i = 1$. Thus

$$C_i = \alpha_2 A_2 + A_i = \beta_2 B_2 + B_i \text{ mod } p$$

The **HNF** form then gives us

$$C_{i,2} = \alpha_2 = \beta_2 \text{ and } C_{i,1} = \alpha_2 A_{i,1} + A_{2,1} = \beta_2 B_{i,1} + B_{2,1} \text{ mod } p$$

which simplifies to

$$C_{i,2} = \alpha_2 \text{ and } C_{i,1} = \alpha_2 A_{i,1} + A_{2,1} = \alpha_2 B_{i,1} + B_{2,1} \text{ mod } p$$

the solution for α_2 is trivially unique (mod p) and gives the result. □

Note that given such an intersection result, i.e of determinant p^2 and the upper-left corner is essentially pId_2, there is at least $p(p-1)/2$ possible combinations of perfect **HNF** of determinant p lattices to obtain the given intersection. The next property we will express will deal with lattices that are not perfect but still has an unique non-zero column, i.e "pseudo-perfect".

Property 15 (Intersection with a single non-zero column).
*Let $A, B \in \mathbb{Z}^{n\times n}$ be full-rank matrices in **HNF** form, and C the **HNF** basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$. Let $n > j > 1$ such that $B_{j,j} > 1$, and*

$$\begin{gathered} \forall i < j,\ A_{i,i} > 1 = B_{i,i} \\ \forall i > j,\ A_{i,i} = 1 = B_{i,i} \end{gathered}$$

Then for $i < j$*, we have* $C_i = A_i$*, and thus* $\mathcal{L}' =< A_1, ..., A_{j-1} >_{\mathbb{Z}} =< C_1, ..., C_{j-1} >_{\mathbb{Z}}$*.*

$$C_j = B_{j,j} \times A_j \ \ mod\ \mathcal{L}'$$
$$For\ k > j,\ C_k = B_{k,j} \times A_j + A_k \ \ mod\ \mathcal{L}'$$

Proof. The first rows staying identical to A's and the head coefficients' values are direct consequences from the previous properties.
We know for a fact that any coefficient before the j-th column is reduced to 0 mod $\mathcal{L}(B)$ as $< B_1, ..., B_{j-1} >_{\mathbb{Z}}$ is $\mathcal{L}([Id_{j-1} \mid 0])$. Therefore for $k \leq j$ we only need to make the columns of C_k reducible to 0 mod $\mathcal{L}(A)$.
Let $v = B_{j,j} \times A_j$. v is trivially reducible to 0 in $\mathcal{L}(A)$ and $\mathcal{L}(B)$, thus $\mathcal{L}(A) \cap \mathcal{L}(B)$. If we reduce v's coefficients modulo $\mathcal{L}'$, we obtain the shortest all-positive vector belonging to a **HNF** thus $C_j = v$ mod $\mathcal{L}'$. Same for $C_k = B_{j,k} \times A_j + A_k$ mod $\mathcal{L}'$. □

While exhibiting a reduction by a lower-rank sublattice can seem expensive, using the **HNF** form to apply a Gauss-Jordan reduction make it inexpensive:

Example 7. A, B, C *three* ***HNF*** *and* $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$*,* $j = 3$*.*

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 4 & 7 & 0 & 0 & 0 \\ 1 & 6 & 1 & 0 & 0 \\ 3 & 5 & 0 & 1 & 0 \\ 2 & 3 & 0 & 0 & 1 \end{bmatrix} \cap \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 3 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 4 & 7 & 0 & 0 & 0 \\ 4 & 2 & 5 & 0 & 0 \\ 0 & 4 & 1 & 1 & 0 \\ 3 & 0 & 3 & 0 & 1 \end{bmatrix}$$

Here $\mathcal{L}' =< A_1 = C_1 = [5, 0, 0, 0, 0],\ A_2 = C_2 = [4, 7, 0, 0, 0] >_{\mathbb{Z}}$*.*
Third vector:
$v_3 = B_{3,3} \times A_3 = [5, 30, 5, 0, 0]$*,* $C_3 = v_3$ *mod* $\mathcal{L}'$*.*
$C_3 = [5, 30, 5, 0, 0] - 4 \times [4, 7, 0, 0, 0] = [-11, 2, 5, 0, 0] + 3 \times [5, 0, 0, 0, 0] = [4, 2, 5, 0, 0]$ *mod* $\mathcal{L}'$*.*
Fourth vector:
$v_4 = B_{4,3} \times A_3 + A_4 = [1, 6, 1, 0, 0] + [3, 5, 0, 1, 0] = [4, 11, 1, 1, 0]$*,* $C_4 = v_4$ *mod* $\mathcal{L}'$*.*
$C_4 = [4, 11, 1, 1, 0] - [4, 7, 0, 0, 0] = [0, 4, 1, 1, 0]$ *mod* $\mathcal{L}'$*.*

So far we have only spoke of computing intersections, but decompositions are also a key argument we will use later so we propose here a simple one based on the property we just described.

Property 16 (Trivial Decomposition of crypto-type q-ary lattices)**.**
Let $A \in \mathbb{Z}^{n \times n}$ *be in* ***HNF****,* $m < n$ *and* $< A_1, ..., A_m >_{\mathbb{Z}} = \mathcal{L}([q \times Id_m \mid 0])$*, and* $< A_{m+1}, ..., A_n >_{\mathbb{Z}} = \mathcal{L}([B \mid Id])$ *for some* $B \in [0, q-1]^{(n-m) \times m}$*. Then:*

$$A = \bigcap_{i=1}^{m} A^{(i)} \ \ where$$
$$For\ diagonal\ coefficients,\ \forall j \neq i,\ A^{(i)}_{j,j} = 1\ \ and\ A^{(i)}_{i,i} = q$$
$$For\ the\ lower\ triangle,\ \forall j \neq i,\ A^{(j)}_{j,i} = 0\ \ and\ A^{(i)}_{j,i} = A_{j,i}$$

Proof. Just apply recursively the previous formula for this special case. This special case has $\mathcal{L}' = [q \times Id \mid 0]$, which transforms "mod $\mathcal{L}'$" into "mod q" to make it easier. □

The next example shows how in those cases the decomposition is trivial.

Example 8. *Here* $m = 4$, $n = 6$.

$$A = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 1 & 2 & 3 & 4 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{bmatrix} = A^{(1)} \cap A^{(2)} \cap A^{(3)} \cap A^{(4)} \textit{ where}$$

$A^{(1)}$, $A^{(2)}$, $A^{(3)}$ *and* $A^{(4)}$ *are successively*

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 \end{bmatrix}$$

We can push a bit further to obtain even more general formulas for every property in both intersection and decomposition, but the resulting computation complexities will probably increase. At some point, it is possible that the standard way to compute **HNF** intersections of full-rank lattices from [MG12] is actually more efficient than classical modular arithmetic. However for the usage we are going to explicit, in our opinion we gave enough properties to justify our future cryptographic choices. There is however the question of how to deal with non-successive different/linearly dependent rows, i.e when the diagonal elements are not in the right place. The next section is an heuristically efficient way to deal with those issues.

3.2 A HNF vision of column permutations

The title of this section is an explicit answer to the interrogation raised by the previous section. A permutation of the problematic rows and columns can solve most issues. First and foremost, we need to recall a trivial but undocumented property.

Property 17 (Transitivity of **HNF** "perfectness" by non-1^{st} row conjugate by S_n)**.**

*Let $S_{n,[a,b]}$ be the set of matrix representing permutations over all rows of indexes within $[a,b]$ on square matrices of size $n \times n$. Then for any $P \in S_{n,[2,n]}$, and $M \in \mathbb{Z}^{n \times n}$, the conjugate of M by P i.e PMP^{-1} is a perfect **HNF** if and only if M is.*

Proof. The first row and first columns are left untouched. $P = P^{-1}$ for any permutation matrix. Thus, if $M \in \mathbb{Z}^{n \times n}$ is a perfect **HNF**, M stripped of its first column and row becomes $M' = Id_{n-1}$. P stripped of its first row and first column essentially is the same permutation as it was already leaving the first row and column untouched. Let us call this this "new" permutation P'. $P'M'P'^{-1} = Id_{n-1}$ if and only if $M' = Id_{n-1}$ thus giving the trivial result. □

While the property applies for all conjugations as long as the first "cross" (i.e row and column of the same index) is left untouched, we precise permutations as operations are trivial.

Example 9. *Conjugating by* $(2\ 4)$*: central symmetry of crosses* 2,4 *(middle* $[(2,2),(4,4)]$*).*

$$\begin{array}{r} \\ \text{row } 2 \longrightarrow \\ \\ \text{row } 4 \longrightarrow \\ \\ \end{array} \begin{bmatrix} 20 & 0 & 0 & 0 & 0 \\ 19 & 1 & 0 & 0 & 0 \\ 11 & 0 & 1 & 0 & 0 \\ 13 & 0 & 0 & 1 & 0 \\ 17 & 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{conjugate by } (2\ 4)} \begin{bmatrix} 20 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 0 \\ 11 & 0 & 1 & 0 & 0 \\ 19 & 0 & 0 & 1 & 0 \\ 17 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the technique to compute **HNF** intersections regardless of perfectness is to generate successive rows per permutation conjugation where the previous properties apply, then compute the **HNF**, and reapply the exact same conjugation. Since in general the biggest entries of the **HNF** are left in the first row, the arithmetical cost of performing operations in the non-first column should be very small, and the coefficients in the first column are reduced by a simple integral modular reduction by the first row.

If we already have the intersection and we wish to obtain a perfect **HNF** of a lattice that is equivalent (in the sense that its geometrical properties are left untouched), then a permutation with a column is worth a try. This gives us the following property, where only a single permutation is needed:

Property 18 (Existence of a transposition for perfect **HNF**).
*Let $A \in \mathbb{Z}^{n \times n}$ in **HNF**. If there exists $\sigma \in S_n$ a permutation of columns such that $\sigma(A)$ admits a perfect **HNF**, then there exists $i \in [1,n]$ such that $\sigma' = (1\ i)$ is sufficient for $\sigma'(A)$ to have a perfect **HNF**.*

Proof. Let σ be such a permutation. Then there must exist a permutation σ' such that $\sigma' \circ (1\ i) = \sigma$ where σ' leaves the first column untouched (note that $i = 1$ is valid). If σ' does not touch the first column, then so does σ'^{-1}, as σ', σ'^{-1} should be both in the group $S_{n,[2,n]}$ (subgroup of S_n). By the previous property, if $\sigma(A) = \sigma' \circ (1\ i)(A)$ admits a perfect **HNF**, then so does $\sigma'^{-1} \circ \sigma' \circ (1\ i)(A) = (1\ i)(A)$. □

Now we just need to pick which value i in $(1\ i)$ to choose. The trivial property below solves the problem.

Property 19 (Stability of the lower-left corner by permutation)**.**
*Let $A \in \mathbb{Z}^{n \times n}$ in **HNF**. For all $n \geq j > i \geq 1$, A' the **HNF** of* $(1\ i)(A)$ *admits* $A_{j,j} = A'_{j,j}$.

Proof. The columns $k \geq j$ are already reduced, therefore do not change. □

Thus, the unique permutation we can try universally is $(1\ n)$. If A does not admit a perfect **HNF** but neither does $(1\ n)(A)$, then no permutation will work. Of course, to reduce complexity cost, we can also choose to pick $(1\ k)$ where k is the biggest integer where $A_{k,k} \neq 1$.

Example 10. *Swap and recompute.*

$$\begin{bmatrix} 33 & 0 & 0 & 0 & 0 & 0 \\ 32 & 5 & 0 & 0 & 0 & 0 \\ 12 & 4 & 1 & 0 & 0 & 0 \\ 31 & 2 & 0 & 1 & 0 & 0 \\ 19 & 1 & 0 & 0 & 1 & 0 \\ 27 & 3 & 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Permute}} \begin{bmatrix} 0 & 33 & 0 & 0 & 0 & 0 \\ 5 & 32 & 0 & 0 & 0 & 0 \\ 4 & 12 & 1 & 0 & 0 & 0 \\ 2 & 31 & 0 & 1 & 0 & 0 \\ 1 & 19 & 0 & 0 & 1 & 0 \\ 3 & 27 & 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Recompute}} \begin{bmatrix} 165 & 0 & 0 & 0 & 0 & 0 \\ 160 & 1 & 0 & 0 & 0 & 0 \\ 64 & 0 & 1 & 0 & 0 & 0 \\ 157 & 0 & 0 & 1 & 0 & 0 \\ 96 & 0 & 0 & 0 & 1 & 0 \\ 138 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We know that if a **HNF** is perfect, then its associated lattice is co-cyclic and thus lies within the class of lattices in which the average-case to worst-case reduction from [GINX16] applies. The first question that comes to mind is, can we actually generate all the co-cyclic lattices this way? The answer is obviously no, as the cryptographic q-ary lattices with prime q are obviously not co-cyclic under any permutation. To test the efficiency of our conversion rate, we tried to apply the permutation to a set of random elements of $[-7, 7]^{n \times n}$ which did not have a **HNF**. The success rate is presented in table 3.1

Dimension	50	80	100	120	150	180	200	300
Sucess rate	0.756	0.726	0.741	0.748	0.744	0.749	0.728	0.737

Table 3.1: Conversion success rate of imperfect into perfect form

Let us recall that [NS15] reported an asymptotic proportion of 85% of co-cyclic lattices as determinant grows. As we discarded all perfect form **HNF**, for our experimentations, we are actually close to sampling most of them.

Now that we know how to compute intersections and attempt to generate perfect **HNF**, we present the overall idea of our work.

3.3 Structure of lattice intersections

The two previous sections showed we can easily compute intersections, and we can easily factorize a lattice into an intersection of co-prime lattices, i.e lattices with co-prime determinants. The natural question that follows is a question that is central to the security patch we aim to apply: if we know the shortest vectors of several prime lattices, can we easily guess the shortest vector of their intersection? The answer is again no, and to explain why, we showcase in the next example a direct consequence of the properties we have previously proven in the case of cryptographic q-ary lattices:

Example 11. *Shortest vectors of intersections in the case of cryptographic q-ary lattices:*

$$A = \begin{bmatrix} 23 & 0 & 0 & 0 & 0 & 0 \\ 0 & 23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 23 & 0 & 0 & 0 \\ 17 & 16 & 12 & 1 & 0 & 0 \\ 12 & 8 & 22 & 0 & 1 & 0 \\ 19 & 21 & 13 & 0 & 0 & 1 \end{bmatrix} = A^{(1)} \cap A^{(2)} \cap A^{(3)}$$

$$\textit{Shortest vector :} \begin{bmatrix} 2 & -1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{array}{ccc} A^{(1)} & A^{(2)} & A^{(3)} \\ \begin{bmatrix} 23 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 17 & 0 & 0 & 1 & 0 & 0 \\ 12 & 0 & 0 & 0 & 1 & 0 \\ 19 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 26 & 0 & 1 & 0 & 0 \\ 0 & 8 & 0 & 0 & 1 & 0 \\ 0 & 21 & 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 23 & 0 & 0 & 0 \\ 0 & 0 & 12 & 1 & 0 & 0 \\ 0 & 0 & 22 & 0 & 1 & 0 \\ 0 & 0 & 13 & 0 & 0 & 1 \end{bmatrix} \\ \textit{Shortest vectors:} & \textit{Shortest vectors:} & \textit{Shortest vectors:} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Intuitively, we know problems on q-ary lattices are hard thanks to Ajtai's work [Ajt96]. But the problem is trivially easy on its overlattice decomposition elements. Therefore, let us define a "new" problem:

Definition 38 (Shortest vector in an intersection).

Let $\mathcal{L}_1, \mathcal{L}_2$ two lattices with determinant d_1, d_2, which shortest vectors are known. Find the shortest vectors in $\mathcal{L}_1 \cap \mathcal{L}_2$.

Note that we call it here "new" as, to the best of our knowledge, it wasn't explicitely stated in the literature. However the problem was probably known to be hard in academic folklore: it is possible that nobody thought it could be worth publishing before. To the best of our knowledge, the only study on the matter was [PS09]: the problem is seemingly hard when relatively short vectors were completely disjoint, but is definitely easier when they had relatively short common vectors. Which gives us the two following problems:

Definition 39 (Common Shortest Vector Problem (**CSVP**)).
Let $\mathcal{L}_1, ..., \mathcal{L}_k$ a set of k lattices with determinant $d_1, ..., d_k$, which share at least a common shortest vector v. Find v.

Definition 40 (Shortest Disjoint Vector Problem (**SDVP**)).
Let $\mathcal{L}_1, ..., \mathcal{L}_k$ a set of k lattices with determinant $d_1, ..., d_k$, such that

$$\forall i \in [1, k], \exists j \neq i, \forall v \in \mathcal{L}_i, \|v\|_2 < \sqrt[n]{d_j} \implies v \notin \mathcal{L}_j$$

find the shortest vectors of $\mathcal{L}_1 \cap ... \cap \mathcal{L}_k$.

CSVP is the problem tackled in [PS09], while **SDVP** seems to be relatively hard. The conclusion of [PS09] was basically to advocate the use of paddings to avoid **CSVP** and have specific instances of **SDVP** instead (adding random noise to common vectors). As far as **SDVP** is concerned, it is unclear where it was previously tackled on in the literature. Clearly the hardness of the latter depends on the number on multiple factors, and heuristically the problem can be seen hard on cryptographic q-ary lattices: for those lattices most **SVP** instances can be converted into a **SDVP** instance as shown in the previous example. This definition of the problem does not apply when a lattice has a prime determinant, but as primes numbers have a vanishing density[a] we do not consider this argument as making **SDVP** necessarily simpler than **SVP**.

For now we are consider the problem for $k = 2$, i.e for the intersection of only two lattices. Another "simpler" problem then, would be given an intersection of two lattices, which one is the "easy" one. In a sense, it could be seen as another "easier" related problem of Decision Learning With Errors (**DLWE**), depending on the instances considered. Here the problem is voluntarily defined in a general sense, to fit as many instances as possible:

Definition 41 (Decision Lattice Family Problem (**DLFP**)).
*Let S_1, S_2 be two lattice families. Given any number of random **HNF** basis of*

[a] The Prime Number Theorem was allegedly demonstrated independently by both Jacques Hadamard [Had96] and Charles-Jean de la Vallée Poussin [DlVP97]. Note that the references does not reflect the date of the findings (1986).

lattices sampled with chance $1/2$ *from either* S_1 *or* S_2*, decide if* $\mathcal{L} \in S_1$ *or* $\mathcal{L} \in S_2$*, or both.*

Clearly, the hardness of the problem depends on the families S_1, S_2 chosen. For example, given the set S_1 of cryptographic q-ary lattices and the set S_2 of co-cyclic lattices, the answers would be trivial. The same can be heuristically said about the set of ideal lattices and the set of non-ideal lattices. However, seemingly non-trivial examples can be constructed. Let S_1 be the set of **HNF** of diagonal dominant matrices, and S_2 the set of random **HNF** with similar determinants according to the distribution of [GM03] but in which we excludes S_1, can we distinguish in polynomial time whether any lattice $\mathcal{L}$ is in S_1 or S_2? The problem seems to be an open question: unlike **DLWE** the problem which compares two specific distributions, to the best of our knowledge **DLFP** is not proven to be NP-hard for any particular dilemma (obviously not counting the Gaussian vs uniform distribution).

3.4 Example application: reinforcing GGH

We now present our application of the previously described theoretical tools to construct a security patch to structural weakness. Here, we fix the structural weakness of **GGH**Encrypt (note this does not concern **GGH**Sign) present in the public key.

3.4.1 Modification of GGH using an intersecting lattice

If we denote P_k the public key and S_k our private key, our modification is to go from $\mathcal{L}(P_k) = \mathcal{L}(S_k)$ to $\mathcal{L}(P_k) = \mathcal{L}(S_k) \cap \mathcal{L}(R)$ where R is a random non-singular integer matrix of dimension n. The modified **GGH** scheme, informally, is:

- $KeyGen_2(n)$. Take a "good" basis $Sk = (D_{Id}) + ([-\mu, \mu]^{n*n})$ of dimension n, compute the **HNF** basis P_k of $\mathcal{L}(S_k) \cap \mathcal{L}(R)$ where R is a random integral matrix of dimension n with a perfect **HNF** with a determinant co-prime to S_k and provide P_k as the public key and keep S_k as the secret key.
- $Encrypt_2(P_k, m)$. Use P_k to encrypt a message m encoded in a small vector, by adding a random vector v of $\mathcal{L}(P_k)$. Outputs $c = m + v$.
- $Decrypt_2(S_k, c)$. Use S_k to decrypt a message the same way as in a classical **GGH**, separating m from v by solving the corresponding **BDD** instance and thus recovering m.

The secret key in our experiments used a diagonal coefficient was $D = \sqrt{n}$ and a low noise of random values in $\mu = 1$ and our security analysis will be based on those parameters. However, we do not see a problem with taking a noise within $\mu = 4$ in the original **GGH** proposal in [GGH97] or when as it was the case when Micciancio applied the use of a **HNF** [Mic01]. Following the work in [Mic01] and [GGH97], we can also choose our messages such that $\|m\|_2 \leq \frac{1}{2} \min \|s_i^*\|_2$ where s_i^* is the i-th vector of the orthogonalized basis obtained from the secret key using the *Gram-Schmidt orthogonialization* process, or simply encode it as a vector in $[1 - \mu, \mu - 1]^n$.

However, the message space we actually used in the original paper was different as we resorted to a padding technique to attempt to reach IND-CCA security. Note that the main contribution of the paper was not to propose a "new" practical scheme, but to use lattice intersections as a mean to hide the exploitable mathematical structure of secret keys, and in this example case, the **GGH** keys. Thus, we will not mention padding techniques in this thesis: those can always be added on top of the underlying hard mathematical problem.

The decryption works as $v \in \mathcal{L}(P_k) \subset \mathcal{L}(S_k)$, thus separating m from v as before. We will discuss security concerns related to this scheme in a next subsection, especially why $\det(R)$ have to be co-prime to S_k and have a perfect **HNF**. In particular, we will show that structural attacks are no longer effective when R is sufficiently large. $\mathcal{L}(P_k)$ no longer admits a diagonally dominant basis $D + R$ therefore the $\mathbf{BDD}_\gamma$ key recovery attack on vectors of D_{Id} is no longer applicable.

Nevertheless, the ratio between the size of the messages we can decrypt and the size of the public key will decrease. We will explicitly express the factor later.

3.4.2 Modified attack on the intersected GGH key

As stated earlier, the structural key recovery attack using $\mathbf{BDD}_\gamma$ on vectors of D_{Id} in $\mathcal{L}(P_k)$ does not work since $\mathcal{L}(D_{Id} + R) \neq \mathcal{L}(P_k)$, but $\mathcal{L}(D_{Id} + R) = \mathcal{L}(S_k) \subset \mathcal{L}(P_k)$. Adapting this attack to $\mathcal{L}(P_k)$ requires finding $\mathcal{L}(S_k)$ in $\mathcal{L}(P_k)$, and then using the structural key recovery attack. We assume the existence of a function Δ which finds the "optimal" overlattice $\mathcal{L}(S_k)$ given $\mathcal{L}(P_k)$ (the overlattice that admits a diagonal basis, experimental data suggests it is indeed the "weakest" integral overlattice).

To the best of our knowledge, the difficulty of recovering $\mathcal{L}(S_k)$ in $\mathcal{L}(P_k)$ is mostly dependent on the values of $\det(P_k)$ and $\det(S_k)$. The efficiency of the modified attack is dependent of the efficiency of the overlattice recovery function Δ.

In this case, we will consider P_k and R to have a perfect **HNF** and S_k to be able to be reduced to a perfect **HNF** as we believe it offers the best security assumptions:

Algorithm 12 Modified Diagonal Dominant Key recovery attack

```
Require: $P_k$, $D$, $\phi$ a $\mathbf{BDD}_\gamma$ solver, $\Delta$ an "optimal" overlattice
Ensure: $S_k$ the secret key
$\mathcal{L} \leftarrow \Delta(\mathcal{L}(P_k))$    ▷ Find $\mathcal{L} = \mathcal{L}(S_k)$ among all integer overlattices of $\mathcal{L}(P_k)$
$S_k \leftarrow D_{Id}$
for $\{i \in [1..n]\}$ do    ▷ Loop on every position of the diagonal
    $r \leftarrow \phi(\mathcal{L}, S_k[i])$    ▷ Find $r$ the difference between $(0, ..., d_i, ..., 0)$ and $\mathcal{L}$
    $S_k[i] \leftarrow S_k[i] + r$
return $S_k$
```

a perfect **HNF** gives a co-cyclic lattice, which is within the family of hard lattices shown in [GINX16]. It also allows us to make heuristic tests to compare with other co-cyclic lattices of the same determinant using directly the distribution given by [GM03].

If we intersect two lattices whose **HNF** basis are not perfect or whose determinant are not coprime, the result will not be perfect (property 15) or common factors will appear (properties 14,13).

In our case, considering $C = P_k$, $A = S_k$ and $B = R$, we have to avoid $\mathcal{L}(S_k)$ to be easily recovered. Property 12 is also interesting for an attacker as it reveals one of the main issue of our approach which we discuss in the next subsection. Let $\omega(M)$ be the number of prime factors counted without multiplicity in the decomposition of $\det(M)$. Then

$$\det(C) = \prod_{i=1}^{\omega(A)} p_i \ , \ \det(B) = \prod_{i=0}^{\omega(B)} q_i \ , \ \det(C) = (\prod_{i=1}^{\omega(A)} p_i)(\prod_{i=1}^{\omega(B)} q_i) \tag{3.1}$$

which means $\omega(C) = \omega(A) + \omega(B)$ and very easily lead to the following property:

Property 20 (Number of possible decompositions for a perfect **HNF**).
*Let C be a perfect **HNF** square matrix of dimension n. The couples $(\mathcal{L}(A), \mathcal{L}(B))$ where A and B are in perfect **HNF** of the same dimension with $\det(A)$ and $\det(B)$ co-prime such that $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$ and $\det(A)\det(B) = \det(C)$ corresponds exactly to the couples $(a > 0, b > 0)$ where a and b are co-primes such that $ab = \det(C)$: there is exactly $2^{\omega(C)}$ possibilities. Each possible solution $(\mathcal{L}(A), \mathcal{L}(B))$ corresponds exactly to $(\det(A), \det(B))$.*

Proof. Trivial using property 12 □

Therefore recovering $\mathcal{L}(A)$ from $\mathcal{L}(C)$, i.e., the complexity of the overlattice distinguisher Δ, is assumed to be at least polynomially equivalent to distinguishing p_i from q_i in equations 3.1. As we work in post-quantum cryptography we assume that factorization problem can be solved in polynomial time [Sho97]. As we explain later, we will purposely choose keys with very smooth determinants, which make factorization easy even on the non-quantum case.

As $\omega(C)$ is lower-bounded by $\omega(A)$ and have no upper bound limit (as we have complete control over $\det(B)$), one might first think that the number of combinations to search is $2^{\omega(C)}$. However, if we assume an oracle that knows what type of lattice to search for, then it is unsafe to assume the attacker has no knowledge of $\omega(A)$. We then assume an attacker have possession of exactly $\omega(A)$ and $\omega(B)$, the number of combinations he has to try is then $\binom{\omega(C)}{\omega(A)} = \binom{\omega(C)}{\omega(B)}$[b]. In practice, the attacker will probably only have an approximation which grows the number of combinations by quite a lot which is a much bigger number depending on how precise the approximation is, but for simplicity we assume he has the exact value.

Hence, if $\omega(C)$ is small or if $\omega(A)$ or $\omega(B)$ is too small relative to the number of possibilities, it might be too easy to recover $\mathcal{L}(A)$, which would nullify the point of our modification. In practice, if we let the scheme untouched as it is, then $\omega(A)$ will most often be very low, as illustrated by the following theorem:

Theorem 4 (Erdös - Kac Theorem [EK40])**.**
Let $\rho(n)$ the number of prime factors of the integer n, then the probability distribution of $\frac{\rho(n)-\log\log n}{\sqrt{\log\log n}}$ is the standard normal distribution.

Experimental data on low dimensions suggest that $\omega(P_k)$ is indeed too low to ensure a reasonable number of combinations. To deal with that problem, we first optimistically assume that without the knowledge of $\det(S_k)$, an attacker will have no choice rather than trying every possible combination stated by the property 20. Our proposed solution to this apparent weakness is to ensure the number of combinations is sufficiently large to make sure it is not feasible to recover $\mathcal{L}(S_k)$. Therefore our target public key should have this form:

$$P_k = \left[\begin{array}{c|ccc} \det(P_k) & 0 & \dots & 0 \\ \hline P_k[2,1] & & & \\ \vdots & & Id_{n-1} & \\ P_k[n,1] & & & \end{array}\right], \det(P_k) = \prod_{i=1}^{\omega(P_k)} p_i$$

$$\forall i \neq j,\ gcd(p_i, p_j) = 1$$

where $\omega(S_k)+\omega(R) = \omega(P_k)$ is large enough to allow a large number of combinations. To achieve this, we must generate S_k and R such that $\mathbf{HNF}(S_k)$ and R have the same form as P_k (Properties 12,20) while controlling $\omega(S_k)$ and $\omega(R)$.

[b] We note the notation C^n_r is often used instead of $\binom{n}{r}$. They mean "n choose r" and equal $\frac{n!}{r!(n-r)!}$

3.4.3 Countermeasure by controlling $\omega(S_k)$ and $\omega(R)$

From the last subsection two properties must arise from our keys if we want our modification to be effective: one is the perfectness of our **HNF** basis, the other the smoothness of our keys' determinants.

First of all, we begin by showing the generation algorithm of the random matrix R. We first choose the determinant $\det(R)$ we want to obtain, and create

$$R = \left[\begin{array}{c|ccc} \det(R) & 0 & \dots & 0 \\ \hline R[2,1] & & & \\ \vdots & & Id_{n-1} & \\ R[n,1] & & & \end{array}\right]$$

such that $\forall i \in [2,n], R[i,1]$ are random integer values in $[0, \det(R)-1]$.

With the knowledge of how to generate R, one simple way to maximize $\omega(P_k)$ and thus the number of possibilities is to increase $\omega(R)$ until we reach the number of required combinations without caring about $\omega(S_k)$. However, this is not wise: we mentioned before that the ratio size of the message over size of the public key is lower in our modified **GGH** than in the original **GGH**. As in both schemes we use the same private key the size of the message m we decrypt remain unchanged.

Let $P_{k\ ori}$ be the public key of the original **GGH** scheme (i.e $HNF(S_k)$). Noting that $Size(P_k) \approx Size(HNF(S_k)) + Size(R) = Size(P_{k\ ori}) + Size(R) \approx Size(P_{k\ ori}) + n\log(\det(R))$. Let c be the factor determining the ratio decrease. Then we have:

$$c \approx \frac{Size(P_{k\ ori})}{Size(P_{k\ ori}) + n\log(\det(R))} = \frac{Size(HNF(S_k))}{Size(HNF(S_k)) + n\log(\det(R))} \tag{3.2}$$

Therefore compared to the original **GGH** the size increase of the public key is solely determined by $\det(R)$. Thus, for the benefit of key size, we choose to have $\omega(R) < \omega(S_k)$. We will discuss later what this implies at the end of this subsection.

We know have to generate S_k. It is a bit harder to obtain a perfect **HNF** of a diagonal dominant matrix with a chosen determinant $\det(S_k)$, but the permutation technique described previously help tackling the issue. Our targeted end result is

$$HNF(S_k) = \left[\begin{array}{c|ccc} \det(S_k) & 0 & \dots & 0 \\ \hline HNF(S_k)[2,1] & & & \\ \vdots & & Id_{n-1} & \\ HNF(S_k)[n,1] & & & \end{array}\right]$$

such that $\forall i \in [2,n]$, $\mathbf{HNF}(S_k)[i,1]$ are integer values in $[0, \det(S_k) - 1]$ and $\det(S_k) = \prod_{i=1}^{\omega(S_k)} p_i$ and $\forall i \neq j$, $gcd(p_i, p_j) = 1$.

In the following, we will illustrate and then explain the way to achieve this: let $S'_k \in \mathbb{Z}^{(n-1)\times(n-1)}$ be a diagonal dominant matrix with diagonal coefficient D whose **HNF** is perfect, and $c \in [-\mu, \mu]^{(n-1)\times 1}$ where μ is also the bound of the noise of S'_k.

We concatenate c on the left of S'_k and compute the **HNF** of the result to obtain S''_k.

$$S''_k = HNF\left(\left[\begin{array}{c|c} c & S'_k \end{array}\right]\right) = \left[\begin{array}{c|c|ccc} a & b & 0 & \dots & 0 \\ \hline \vdots & \vdots & & Id_{n-2} & \end{array}\right]$$

If $a = S''_k[1,1]$ and $b = S''_k[1,2]$ are not co-prime we retry with a different column c until those two values are co-primes. Once they are co-prime we compute u, v such that $|ua - bv| = \det(S_k)$, ensuring $\det(S_k)$ is co-prime with either a or b (for simplicity we assume b is co-prime with u) , such that

$$2d \times \det(S'_k) > \det(S_k) > d \times \det(S'_k) \text{ and } \det(S_k) \nmid \det(S'_k) \tag{3.3}$$

The probability of having S'_k being perfect and a, b being co-prime is experimentally 1/2. This goes in line with the numbers of table 3.1 to obtain a perfect **HNF**, and the chance of two random integers being coprime according to [HW38][c] are around $6/\pi^2 \approx 61\%$ $(0.8 \times 0.6 = 0.48)$.

We create a line l of n entries with $l[1] = v$ and $l[2] = u$, reduce l with Babai's nearest plane algorithm and concatenate the result l' to c and S'_k as shown below to obtain S_k as we wanted, which is still a diagonal dominant matrix relatively close of parameters d and μ and possess a perfect **HNF**.

$$l = [v, u, 0, ..., 0],\ l' = NearestPlane(l, S''_k)$$

$$S_k = \left[\begin{array}{c|ccc} l'[1] & l'[2] & \dots & l'[n] \\ \hline c & & S'_k & \end{array}\right]$$

And the **HNF** of S_k is perfect, since:

[c] The result is attributed to Euler in 1735 but we lack a reference. Nevertheless, the one we are giving contains a proof: the theorem is numbered 332 at the 269-th page of the 4th edition

$$HNF(S_k) = HNF\left(\left[\begin{array}{c|ccc} l'[1] & l'[2] & \dots & l'[n] \\ \hline & & & \\ c & & S'_k & \\ & & & \end{array}\right]\right) = HNF\left(\left[\begin{array}{c|c|ccc} v & u & 0 & \dots & 0 \\ \hline a & b & 0 & \dots & 0 \\ \hline \vdots & \vdots & & Id_{n-2} & \end{array}\right]\right)$$

and because we ensured a and b co-prime, $|ua-bv| = \det(S_k)$ and u and b co-primes:

$$HNF(S_k) = \left[\begin{array}{c|ccc} \det(S_k) & 0 & \dots & 0 \\ \hline & & & \\ \vdots & & Id_{n-1} & \\ & & & \end{array}\right]$$

Now that we know how to create S_k and R with chosen determinants, we must decide how to choose $\det(S_k)$ and $\det(R)$ to obtain secure values of $\omega(S_k)$ and $\omega(P_k)$. This is how we suggest to do it and use in most experimentations. We first generate D_d as we see fit (the same matrix used when generating S_k), and we fix a prime we call a “scaling factor” that must be prime to $\det(S'_k)$, then enumerate all primes from a certain point (at least strictly greater than the “scaling factor”) and choose to pick them randomly until their product is bigger than S'_k (and prime to $\det(S'_k)$). We denote that set of primes S_p. We then take as much factors of S_p as we see fit to construct $\det(R)$, and keep the rest and multiply the remaining product by a power of the scaling factor to respect the bound given by equation 3.3. Suppose that the scaling factor is given away for free, the number of combinations an attacker has to search is indeed $\binom{|S_p|}{\omega(R)}$.

Overall, the generation of keys is done in polynomial time: the computations of the **HNF** and Babai’s Nearest Plane algorithm are the most time consuming operations and they both run in polynomial time.

We present in table 3.2 the minimum ratio $\omega(R)/\omega(P_k)$ required to go over some 2^λ combinations in total using the algorithm we just described, and to ensure $\omega R < \omega Sk$ we use a scaling factor of 2, enumerating and choosing all primes from 3 until the product S_p is bigger than $\det(S'_k)D$, assuming a diagonal coefficient of $D = \sqrt{n}$ where n is the dimension and getting an average determinant of $n^{n/2}$ for S_k. When S_p is not sufficiently large, then we put the symbol -.

Table 3.3 shows the different results obtained when enumerating from 2741 which is the 400^{th} prime and then his successive primes.

We put a blank entry at $\lambda = 160$ for dimension $n = 800$, but we are actually very

λ	300	400	500	600	700	800
80	34/87	24/115	21/142	19/168	18/195	17/222
100	-	37/115	29/142	26/168	24/195	23/222
120	-	-	42/142	35/168	32/195	29/222
140	-	-	-	48/168	41/195	37/222
160	-	-	-	69/168	53/195	47/222

Table 3.2: Number of primes required to achieve 2^λ, starting from 3

λ	200	300	400	500	600	700	800
80	-	-	-	28/96	23/119	21/141	19/164
100	-	-	-	-	35/119	30/141	27/164
120	-	-	-	-	-	42/141	36/164
140	-	-	-	-	-	-	49/164
160	-	-	-	-	-	-	-

Table 3.3: Number of primes required to achieve 2^λ, starting from 2741

close with $\log_2(\binom{164}{82}) > 159.99$. If we want to enforce $\omega(R) < \omega(S_k)$ while using the same algorithm, we only need to increase d to a bigger value, increasing $\det(S_k)$ and thus $\omega(P_k)$. The table 3.4 shows the minimum amount of primes to put in S_p, which means the minimum value of $\omega(P_k)$ (+1 if we count the scaling factor) to achieve a number of combinations strictly superior to 2^λ.

λ	80	100	120	140	160	180	200	220	240
$\omega(P_k)$	84	104	124	144	165	185	205	225	245

Table 3.4: Number of primes $\omega(P_k)$ required to achieve 2^λ

As a “rule of thumb”, to achieve a number of combinations of 2^λ we require $|S_p| = \omega(P_k) + 1 > \lambda + 4$ for $\lambda \in [1, 250]$. We can deduce that in practice the number of combinations will always be as high as we need it to be, either by increasing d the diagonal coefficient of our secret key S_k or by adding factors to R. Therefore security concerns do not lie in the number of combinations $(\mathcal{L}(S_k), \mathcal{L}(R))$ any longer.

3.4.4 Key sizes comparisons and perfect HNF

First of all, we would like to stress that the following comparison only aims to show that our obfuscation technique applied to **GGH** is not impractical as far as storage is concerned. **LWE**-based cryptosystems usually rely on stronger security assumptions than our proposal and other cryptosystems like **NTRU** have been studied for a much longer time: the lack of literature on lattice intersections do not let us claim the same level of confidence in security assumptions.

Dimension	q	Public Key size per User
128	2053	1.8×10^5
192	4093	2.9×10^5
256	4093	4.0×10^5
320	4093	4.9×10^5

s \ n	125	150	175	200	225	250	275	300	325	350	375	400
20	6.1×10^4	8.8×10^4	1.2×10^5	1.6×10^5	2.1×10^5	2.6×10^5	3.2×10^5	3.9×10^5	4.6×10^5	5.3×10^5	6.2×10^5	7.1×10^5
40	6.5×10^4	9.4×10^4	1.3×10^5	1.7×10^5	2.2×10^5	2.7×10^5	3.3×10^5	4.0×10^5	4.7×10^5	5.5×10^5	6.3×10^5	7.3×10^5
60	7.7×10^4	1.0×10^5	1.4×10^5	1.8×10^5	2.3×10^5	2.8×10^5	3.4×10^5	4.1×10^5	4.9×10^5	5.6×10^5	6.5×10^5	7.5×10^5
80	-	-	1.6×10^5	2.0×10^5	2.4×10^5	2.9×10^5	3.6×10^5	4.3×10^5	5.0×10^5	5.8×10^5	6.7×10^5	7.6×10^5
100	-	-	-	2.2×10^5	2.6×10^5	3.1×10^5	3.8×10^5	4.4×10^5	5.2×10^5	6.0×10^5	6.9×10^5	7.8×10^5

Table 3.5: **LWE** key sizes from Lindner and Peikert on top, key sizes from perfect **HNF** and intersections at the bottom ($n = dimension$, $s = \log_2\left(\frac{\omega(P_k)}{\omega(S_k)}\right)$)

Here we compare in table 3.5 our public key sizes to to the public key sizes (in bits) presented in Lindner and Peikert's work for **LWE**-based encryption [LP11] and the key sizes we obtain with intersections (in average). We only compare with the size of their keys per user, and not their full key which is already much bigger. Note that unlike $q-$ary lattices the value of the determinant is not set at the key generation, but usually do not stray away from each other by too much bits. To compute the key size of a perfect **HNF**, one just have to compute look at the number of bits of the determinant, and multiply it by the lattice dimension. For dimension n, we use $\sqrt{n}$ as the diagonal coefficient and $[-1, 1]$ as noise interval.

As we see, their partial public key is only smaller than our full public keys for higher dimensions. Furthermore, the techique used in their scheme is a very clever way to delegate part of the key to a trusted source or to the user that is an instance from an abstract system presented by Micciancio [Mic10], while he scheme we present uses the basic setup from the **GGH**Encrypt cryptosystem [GGH97] with almost no modification. It might be possible that in the future that such techniques become available for classical random lattices (and most of them admitting a perfect **HNF**), leading to better key sizes per user for higher dimensions.

Perfect **HNF** also allows us to improve the encryption scheme over the original **GGH** as explained in [Mic01]. Instead of sending a vector $c = m + v$ where $v \in \mathcal{L}$ is random, we can choose v such as $v = (c_1, 0, ..., 0)$ i.e., only a single integer of size $\det(\mathcal{L})$ in average will be sent. This is done by reducing the message m by the Gauss-Jordan method (see example 2) using the public key, which is a perfect **HNF**. Furthermore, the security is not affected. Since the transformation will give the same result as m for any $m' = m + v'$ with any $v' \in \mathcal{L}$, breaking this transformation will break all schemes which use $\mathbf{BDD}_\gamma$ as a security assumption, assuming of course our new keys are secure. The decryption is left unchanged, since the only difference is that $c = m+v'$ where $v' \in \mathcal{L}$ is not random anymore. Lindner and Peikert also presented in their paper their different ciphertext size for messages of 128 bits. The smallest ciphertext size they presented was for $n = 128$ and $q = 2053$, and it has the same size of our average determinant size (hence, in our case, cipertext size) for a lattice of dimension $n = 475$ and combinations security $s = 100$.

Generating public keys, however, is slow, as it involves computing **HNF**. However, research to compute **HNF** nowadays are mostly done for random matrices [PS10, PS13] and not targeted at structured matrices, especially ones that arise in cryptography. A rebound of interest towards **HNF** in the community might lead to similar improvements in the future.

On a side note, the fact that we control the determinant of our key also allows us to use modular inverse for **GGH**. To the best of our knowledge, modular inverses are usable and we conducted small experiments that seemed to verify that claim.

3.5 Security of the reinforced GGH

The security assumptions relies on two important points:

- **Assumption 1**: Recovering the "optimal" integer overlattice is hard. (Δ is polynomial on the number of combinations).
- **Assumption 2**: The underlying $\mathbf{BDD}_\gamma$ problem is hard (on our specific keys).
- **Assumption 3**: The underlying modular knapsack problem is hard (on our specific keys and messages).

Like every cryptosystem, even when based on a hard problem, what we use is specific instances of a hard problem, which might not be as hard as the original problem. Therefore we discuss in the following the special structure which arise from our scheme. Note that because the main idea is to change the public key without changing the secret key while keeping the encryption/decryption process unchanged, the message space is left unchanged along with the $\mathbf{BDD}_\gamma$ problem from the previous iteration from Micciancio [Mic01], except the lattice has a slightly bigger determinant.

Furthermore, our first assumption is actually very pessimistic. Micciancio's scheme has not been broken asymptotically, and up to this day basis reductions techniques are still not well understood enough to predict given a **HNF** how easy it would be to reduce the corresponding lattice. It is therefore possible that in the future further theorical studies will allow us to strongly reinforce our first assumption. Our particular structure (perfect **HNF**) also allows to convert our instance of the $\mathbf{BDD}_\gamma$ problem to an instance of a single general modular knapsack problem with very smooth moduli, therefore our third assumption is that our instances of modular knapsack with smooth determinant are hard, which can be seen as a multiple modular knapsack with coprime modulis.

3.5.1 Smoothness of determinant

To discuss the problem of the smoothness of the determinant, we assume the existence of a polynomial time solving oracle which can determine if a combination of primes lattice is the correct and output a weak basis out of it if yes. In other words, we assume an attacker can solve the **DLFP** we defined previously (see definition 41) in constant time given a combination of primes, and also recover the secret key of a **GGH** in constant time given a **HNF** of the lattice, which is clearly an exaggeration of an attacker's capacity considering the current state of the art. Nevertheless, assuming the "weak integer overlattice" detection oracle runs in constant time, the latter security is bounded by $\binom{\omega(P_k)}{\lceil \omega(P_k)/2 \rceil}$, which as we discussed on the previous section is not a problem in high dimensions as $\omega(P_k)$ grow big, as in our tests we never reach that bound (table 3.4).

In a less exaggerated assumption, we also assume that the attacker does not have an algorithm that permits to eliminate prime overlattices one by one until only factors of S_k remains, or something easy to decipher with which would lead to an easier solution that searching through every possibility, which we think is reasonable as our experimentations could not distinguish any overlattice from random lattices. If there was such an algorithm, then this might apply to any random lattice and lowers the overall security of all lattice problems. However, it is possible that the problem is much easier depending of the kind of secret keys we actually use and what we intersect it with, as we previously mentioned when defining **DLFP**.

The only attacks based on overlattices according to the best of our knowledge were one from Becker, Gama and Joux [BGJ13] and one from Gama, Izabachene, Nguyen and Xie [GINX16] even then the way their overlattices are generated is very different and does not search for integer overlattices specifically. Aside from the overlattices consideration, there is also no attack in the best of our knowledge that makes use of a smooth determinant on random lattices, and other popular schemes such as **NTRU** [HPS98] and Ring-Learning With Errors (RLWE) [LPR10] also rely on lattices with smooth determinants ($q-$ary lattices have naturally smooth determinants, but again their factorization differ and they are not co-cyclic).

Under all of those assumptions, the total security provided by our enhancement is either solving the problem directly on the public key, either finding the right combination multiplied by the time of running a detection oracle (find which integer overlattice is weak), or working in a properly chosen non-integer overlattice of a large enough volume. The latter possibility, which could seem the most efficient, is in our opinion not-effective: to the best of our knowledge our system do not hold a particular weakness towards this approach compared to any other random lattice as our public key seem to hold the same structure as far as our heuristic experimentations on **HKZ**-reduced basis are concerned (see next subsection).

We stress that finding the shortest vector for a small prime overlattice does not help solving the problem in the ring product in general as shown in the example 11. It is in fact still a research problem to be able to compare two numbers given their decompositions over a ring product without computing them back, as it is the main issue with Residue Number System (RNS) (RNS for cryptography is an old and still active research topic [BP04, GLP$^+$12, BI04, BEM16]). This is even more problematic when comparing vectors.

3.5.2 Perfectness of basis and primality between factors

Due to Goldstein and Mayer's work [GM03], taking a perfect hermite normal form matrix with a random prime determinant can be considered as taking random lattice. As we are intersecting a lot of lattices of this type, with different prime determinants which result in a perfect **HNF**, we are comparing our results with other perfect **HNF** bases with the same determinant and random entries. Since our experimental results show that 80% of basis generated with coefficients from bounded entries admits a perfect **HNF** or are equivalent to one by permutation of columns (see table 3.1), we believe the comparison to be fair.

This is further reinforced due to recent works from Nguyen and al [NS15], used shortly after by Gama and al [GINX16] to make their generalized worst-case to average-case reduction which allows us to use a much more general lattice form, namely the co-cyclic lattices, and therefore very different from those q-ary lattices first introduced by Ajtai [Ajt96]. And in practice, Chen and Nguyen's work [CN11] on Darmstadt's lattice challenges lead us to think it is easier to solve **SVP** in a q-ary lattice than in a random lattice of large volume, therefore in term of basis recovery attacks the perfect **HNF** could be actually more desirable.

Furthermore, given a finite set of prime factors on the diagonal, the perfect form gives the hardest challenge, as it is harder to guess a large number of factors in a single position rather than a small fixed amount in multiple positions (given the same total amount of primes), provided we could make sure it wouldn't be able to transform into a perfect **HNF** by permutation (if having a perfect **HNF** becomes a weakness, which again is unlikely given the work in [GINX16]).

Having factors prime to each other not only ensures an easier perfect **HNF**, but also avoid giving information beneficial to the attacker (as stated previously, using property 13). As stated before, having a perfect **HNF** is also desirable for when managing keys.

3.5.3 Shortest Vector and Basis Structure

Now the problem is to determine whether or not our approximate instances of **SDVP** (see definition 40) are hard. We present the result of experimentations for intersecting diagonal dominant type matrices with a random one with a perfect **HNF** form below. We chose 3 as our scaling factor, allowing our perturbation matrix to have a measurable determinant as a power of 2. To determine the impact of the random intersection $\mathcal{L}(R)$ over P_k's resistance against enumeration and classical lattice reduction techniques compared to random lattices [Sch10], we also observe the distribution of coefficients by solving **SVP** on small dimension (40), comparing them to the random case (every public key sample is tested with another random lattice with the same determinant and dimension). It seems like after reaching a size of 32 bits for $\det(R)$ there is almost no difference between P_k or random lattices of the same determinant, and as the difference tends to decrease very rapidly when $\det(R)$ increases, we scale the graphs to the extremas (Figure 3.1).

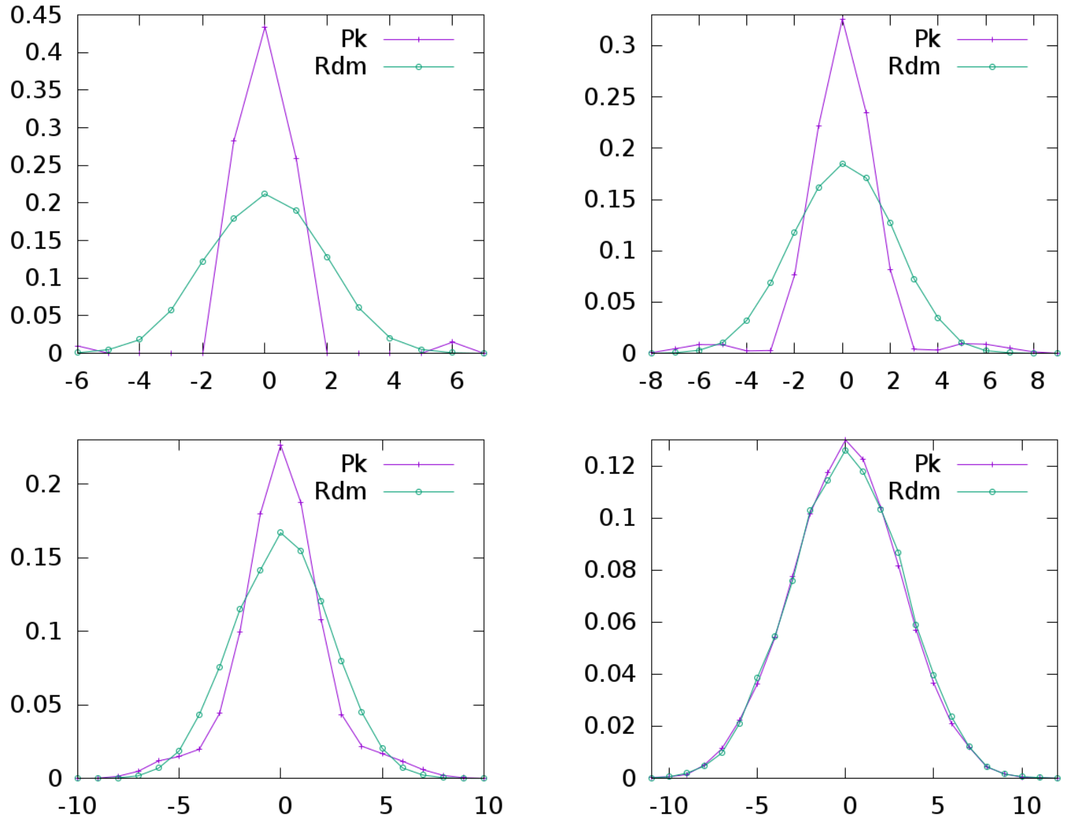


Figure 3.1: Shortest vector's coefficients distribution. $\det(R) = 1, 2^8, 2^{16}, 2^{32}$

However, the obfuscation of $\mathcal{L}(S_k)$'s structure is not exclusive to $\mathcal{L}(S_k)$' shortest vectors. We show with the measures of the condition number (table 3.6,3.7,3.8) that the obfuscation work on whole **HKZ**-reduced basis of $\mathcal{L}(P_k)$. In that regard, we compare condition numbers (CN) with the max norm. The test is done in dimension 30, 40 and 50 with over 20 matrices per dimension and 20 witnesses per new determinant, the diagonal dominant type being having noise in $[-1, 1]$ with the diagonal being $\lceil\sqrt{dim}\rceil$. We only choose matrices with a perfect form. Every matrix computed has been **HKZ** reduced.

$\det(R)$	1	2^4	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}
Avg CN (inter)	55.33	158.48	199.38	234.25	260.06	263.49	262.40	268.40	263.10
Avg CN (rdm)	269.41	273.39	272.88	273.99	273.74	267.13	267.31	271.71	272.65

Table 3.6: Condition number of $\mathbf{HKZ}(P_k)$. Dimension 30, $\log_2(\det(S_k)) \cong 79$

According to our experimental results there is little influence on increasing the size of the perturbation over 32 bits, as we get very close to the same condition num-

det($\boldsymbol{R}$)	1	2^4	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}
Avg CN (inter)	69.52	304.18	400.67	529.02	620.70	691.42	709.63	748.11	769.18
Avg CN (rdm)	755.89	760.90	805.09	755.15	789.46	786.06	770.28	760.09	786.72

Table 3.7: Condition number of $\mathbf{HKZ}(P_k)$. Dimension 40, $\log_2(\det(S_k)) \cong 113$

det($\boldsymbol{R}$)	1	2^4	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}	2^{36}	2^{40}	2^{44}
Avg CN (inter)	78.98	524.19	636.06	837.49	1087.47	1426.19	1616.31	1682.42	1795.90	1830.14	1870.03	1871.08
Avg CN (rdm)	2038.68	1976.51	1993.44	1952.17	1944.57	2021.48	2074.12	1991.93	1964.16	1979.45	2000.98	1998.67

Table 3.8: Condition number of $\mathbf{HKZ}(P_k)$. Dimension 50, $\log_2(\det(S_k)) \cong 151$

ber as a random **HKZ** reduced matrix with the same determinant. This reflects the results we have when measuring the distribution of shortest vectors' coefficients values. This means that obtaining a good basis of of the public key will allow to decrypt nearly as well as with a good basis of a random matrix, being very sensitive to noise. Therefore the only factor to consider is the number of primes, and as we grow larger in dimension, we will obviously take much bigger primes, ending up in a noise with a determinant size of over 32 bits. We can then take small primes to minimize the lattice gap (as we will measure below).

What we need to consider is the possibility to distinguish the different overlattices very easily (which would nullify our improvement of **GGH**), or possible to know if by intersecting different overlattices we could determine if we are getting closer to the right combination or not. Therefore, we compare condition numbers of S_k and R's overlattices, the intersection of S_k's overlattices and R's overlattices separately or mixed together (with all matrices being **HKZ** reduced) to **HKZ** reduced basis of random lattices of the same respective determinant for every test. The result is that the overlattices themselves seem to be indistinguishable from random cases.

Since removing several factors of S_k randomly does seem to make the problem harder (in the sense that the resulting lattice looks more random than keeping all factors), this observation should also hold for non-integer overlattices. For now, it does not seem that an attack on a random overlattice would be more effective.

3.5.4 Message Recovery Attacks

We now measure the influence of R on P_k against message recovery attacks, and we will use the lattice gap in that regard. We assume the attack used is the popular heuristic transformation from **CVP** to **SVP** by attacking an extended basis B (see 2.5.9). What we need then is closely enough approximate values of $\lambda_1(B)$ and $\lambda_2(B)$. In that regard, we consider $\lambda_1(B)$ the size of our message m. With maximum decryption capacity, we have:

$$||m\ S_k^{-1}||_\infty \leq ||m||_\infty ||S_k^{-1}||_\infty \leq \frac{1}{2}$$

As we take the best value possible:

$$||m||_\infty = \frac{1}{2||S_k^{-1}||_\infty}$$
$$\lambda_1(B) = ||m||_2 \approx \frac{\sqrt{n}}{2||S_k^{-1}||_\infty}$$

and we assume $\lambda_2(B) \approx \sqrt{\frac{n}{2\pi e}} Det(P_k)^{1/n}$ with the Gaussian Heuristic. It is the best approximation we can use, as we do not know the length of the shortest vector in the public key. Therefore the lattice gap is

$$\delta(Pk, S_k) = \left(\frac{Det(P_k)^{1/n} \times 2||Sk^{-1}||_\infty}{\sqrt{2\pi e}}\right)$$

It is a bit different than most schemes, as our public key does not represent the same lattice than our private key. The first remark we do when looking at the formula is that the gap increase as $\det(R)$ increase, which might give us a good reason to carefully manage $Size(\det(R))$ besides key size considerations.

Here, we present our experimental results for the simulations on computing the root lattice gap (with primes starting from 3, scaling factor 2). From the earlier work from Gama and Nguyen [GN08], the problem is solvable as soon as the lattice gap is lower than a fraction of the Hermite factor. We observe the evolution of the lattice gap for different number of combinations over increasing dimensions, and it appears that increasing the number of combinations have a lower impact as we increase dimensions, thus a high value for $\det(R)$ is not a problem as dimensions increase.

As the constants mentioned in Gama and Nguyen's work [GN08] depends of the algorithm used and the lattice structure, we scale the curve with factors 0.50 and 0.20. We can observe that, in the pessimistic assumption of a constant of 0.20 (which is not the worst since [GN08] mentions a factor of 0.18 for **BKZ**-20), we do not reach the optimal factor of 1.005 (considered by many to be impossible to reach without proper structure in dimension 500 [GN08]) even past dimension $n = 850$ (figure 3.2).

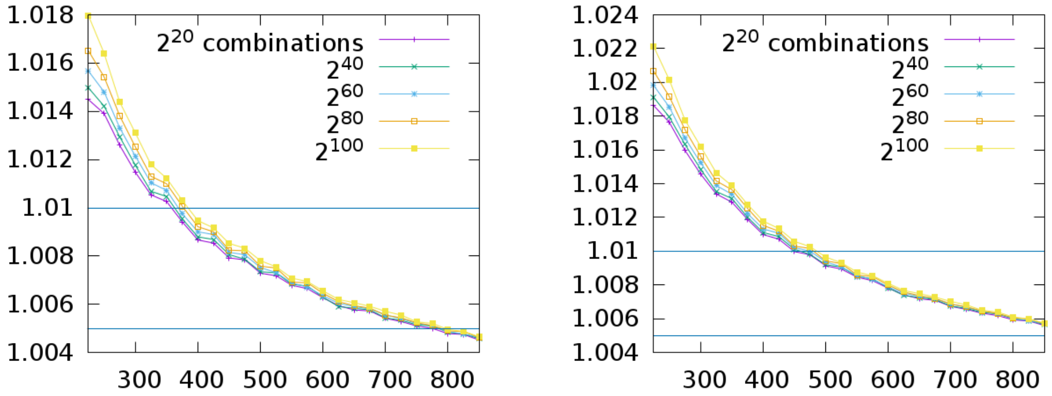


Figure 3.2: Root Lattice Gap Evolution from dim 225 to 850, $C = 1/2$ on the left and $C = 1/5$ on the right

3.6 Open questions and related work

3.6.1 On Ideal Lattices

While we did not experiment much on ideal lattices, we can still assert some (trivial) results:

Property 21 (Intersection of two ideal lattices contains an ideal lattice)**.**
For two ideal lattices $\mathcal{L}(p, f), \mathcal{L}(q, f)$, $\mathcal{L}(\mathrm{lcm}(p, f), q) \subseteq \mathcal{L}(p, f) \cap \mathcal{L}(q, f)$

Proof. $\mathrm{lcm}(p, q) \in \mathcal{L}(p, f) \cap \mathcal{L}(q, f)$ and the quotient operations still hold. □

Experimentally speaking the equality is often met. Results are unknown to us so far when intersecting two ideal lattices of different quotients.

Property 22 (Ideal perfect **HNF** intersections are only made of Ideal perfect **HNF**)**.**
*Let $\mathcal{L}_1, \mathcal{L}_2$ be two lattices admitting perfect **HNF** as basis. Then $\mathcal{L}_1 \cap \mathcal{L}_2$ is an ideal lattice admitting perfect **HNF** as a basis if and only if $\mathcal{L}_1, \mathcal{L}_2$ are both ideal lattices.*

Proof. Combination of the facts given by the property of the intersection of perfect **HNF** and the fact that for a perfect **HNF** of an ideal lattice, the compressed representation always work. The compressed representation is also always a basis of an ideal lattice. □

We would note that the intersection of an ideal lattice to a random lattice is thus not an ideal lattice. The same with applying a permutation to an ideal lattice as it destroys the root structure.

Property 23 (Permutation of the ideal lattice structure)**.**
*Let S_n be the set of permutations of n columns. In general, if H is the **HNF** basis of an ideal lattice and $\sigma \in S_n$, $\mathcal{L}(\sigma(H))$ is usually not an ideal lattice.*

Proof. The result of the permutation loses the root structure in the general case. □

Using an intersection of permuted ideal lattices could have some cryptographic uses. The potentially interesting follow-up however is how to exploit this as a cryptanalytic tool: intersections were used previously for cryptanalysis [PS09] and it is unknown if new attacks could arise when using specifically crafted intersections. In particular, most cryptosystems relying on ideal lattices (and module lattices) rely on consistently using the same quotient polynomial, and it is unclear if that fact is exploitable.

3.6.2 Recovering a vanishing structure

As a rule of thumb, we usually say that the more "zeroes" a key has, the more decryption capacity it can offers, and the more computation efficiency too. But this goes both ways: cryptanalytically, the more "zeroes" a key has, the easier it is to attack and lattice-based crypto is not an exception [Alb17]. A question is thus how efficiently can specific combinations of intersections and permutations of lattices can hide "zeroes", and if identifiable prime lattices overlattices can be detected by their sparsity.

This remark however is unknown given several other structures: we know that ideal lattices are hard to hide, as their **HNF** is too structured and the same can be said about module lattices to a certain extent. But the answer could change depending of the structure considered: tensor-products [FS99], rotations [Slo83], etc...

3.6.3 Group representation of Gama-Izabachene-Nguyen-Xie

In this chapter we shown that intersections of lattices can be used not only as a cryptanalysis tool as in [PS09] but also as a cryptographic construction. Intuitively, we can easily see the parallel between our work and [GINX16, BGJ13]: intersecting two lattices of co-prime determinant is a construction of two groups to another. If we intersect two full-rank lattices $\mathcal{L}_1, \mathcal{L}_2$ of dimension n of co-prime determinant p, q respectively, then using the same notation as in [GINX16], the intersection is an application $\Phi_{p,q}$ such that

$$\Phi_{p,q} : \mathbb{Z}^n/\mathcal{L}_1 \times \mathbb{Z}^n/\mathcal{L}_2 \to \mathbb{Z}^n/(\mathcal{L}_1 \cap \mathcal{L}_2)$$

And in the case of perfect **HNF** lattices given $\text{GCD}(p, q) = 1$, $\Phi_{p,q}$ is a bijection

$$\Phi_{p,q} : \mathbb{Z}_p^n \times \mathbb{Z}_q^n \to \mathbb{Z}_{pq}^n$$

Not all lattices $\mathcal{L}$ such that $\mathbb{Z}^n/\mathcal{L} \simeq \mathbb{Z}_p^n$ have hard problems, but in average they should have. For $\mathcal{L}_2 \subset \mathcal{L}_1$, does a weak instance in $\mathbb{Z}^n/\mathcal{L}_1 \simeq \mathbb{Z}_p^n$ necessarily produces weaker instances in $\mathbb{Z}^n/\mathcal{L}_2 \simeq \mathbb{Z}_{pq}^n$?

From a cryptanalytic point of view, we experimentally showed some weak instances (i.e **GGH**) can trigger weaker instances, however the weakness is less apparent, and it is still unclear if structural lattice reduction can be tweaked to exploit it or even find more apparent weaknesses.

From a cryptographic point of view is there any provable injection from weak instances into $\mathbb{Z}^n/\mathcal{L} \simeq \mathbb{Z}_p^n$ to a provably hard or truly random instance of $G \simeq \prod_{i=0}^{k}(\mathbb{Z}_{p_i}^n)^{m_i}$? We saw on the q-ary case this is doable, but however the surjection was trivial and we would need one-way functions with trapdoor. The GCD descent property showed us q-ary lattices could be of use, but for key sizes purposes other families would be more profitable to study.

Summary and conclusion

We presented a previously unused structure of lattice-based cryptography and applied it to the reinforcement of **GGH**. As the arising problems are relatively undocumented, further work would be necessary to either exploit that new structure for enhancing other schemes, or extend the existing knowledge in cryptanalysis.

Chapter 4

Diagonal-Dominant Lattice-Based Signatures

We submitted to the NIST PQC Standardization process a lattice-based signature-scheme based on an old idea of [PSW08], namely the Diagonal Reduction Signature digital signature scheme **DRS**. This chapter is mainly a paraphrasing of the work we have published on this subject, in particular [SPS20]. Before we present our contributions, we must present the context that led to it.

The theoretical framework of [PSW08], which we will denote **PSW** as per its authors Plantard-Susilo-Win, seemed to have been unchallenged for more than 10 years. Personal feedback was that various people tried to break it but failed. Sadly, there was no paper mentioning trying and failing. The academic silence on the matter could mean several things:

- The problem is so trivial to break commenting on it is "a waste of time".
- The problem is hard to break but so unusable building on it is "a waste of time".
- Trivial and unusable. The worst of both worlds and we were blind to it.
- The paper was not well-explained, so nobody understood and moved on.
- Influential committee members have beef with the original authors of the paper.

We hopefully believed it was none of the above (but had some doubts with the last one). Instead, we believed PKC, i.e *The International Conference on Practice and Theory in Public Key Cryptography* (and not the more famous *Protein kinase C*), had good reviewers and an internationally well-known program committee of excellent quality in exactly 2008 (we say nothing of other years).

Thus, the decision was made to present an instantiation of the **PSW**-signature framework as a Round 1 Candidate for the NIST PQC Standardization process. (**PSW** stands for Plantard-Susilo-Win, as the authors of the original paper [PSW08], while **DRS** stands for Diagonal Reduction Signature, a signature scheme based on the **PSW** framework and proposed to the NIST.)

4.1 The theoretical framework of Plantard-Susilo-Win

4.1.1 Before PSW: lattices for number representation

Before we present **PSW**, we will briefly give some hindsight about "number systems", i.e ways to represent a number. The reason might not be apparent, but we hope a few examples will actually help understand the core ideas behind **PSW** and **DRS**. We are not going to talk about RNS or the CRT which are famous number systems, but clearly irrelevant for what we present in this chapter. Instead, we are going to talk about the very basic representations of numbers.

Suppose we want to represent a number x in base k, such that $x < k^n$. Then the number has the unique following representation:

$$x = x_0 + x_1 k + x_2 k^2 + ... + x_n k^n \text{ such that } \forall i,\ x_i \in [0, k-1]$$

Basically, the role of k is mostly to determine the number of symbols used, and the positions $1, k, k^2, ..., k^n$ are written from increasing power from left to right for a simple representation when writing the number

$$x = \text{“}x_0 x_1 k x_2 ... x_n\text{”}$$

Using $k = 10$ gives us the arabic numerotation most of us use today in science. but then, what if we decide to strip the condition "$\forall i,\ x_i \in [0, k-1]$"? The representation is then obviously not unique anymore:

$$x = (x - \lfloor x/k \rfloor) + \lfloor x/k \rfloor k =$$
$$(x - (\lfloor (x - \lfloor x/k^2 \rfloor)/k \rfloor + \lfloor x/k^2 \rfloor k^2)) + \lfloor (x - \lfloor x/k^2 \rfloor)/k \rfloor + \lfloor x/k^2 \rfloor k^2 = ...$$

Most informed people would see here a reversing of the table euclidean division. However we are choosing another representation: we can also represent this phenomena by a vector. In the following example we reverse the order, putting the highest degree on the left:

$$1851 \simeq [1,8,5,1]$$
$$1851 = 1 \times 10^3 + 0 \times 10^2 + 85 \times 10^1 + 1 \times 10^0 \simeq [1,0,85,1]$$
$$1851 = 0 \times 10^3 + 18 \times 10^2 + 0 \times 10^1 + 51 \times 10^0 \simeq [0,18,0,51]$$

We can see here, that all numbers are obtained by linear combinations by the vectors of the following matrices:

$$B_{10} = \begin{bmatrix} -1 & 10 & 0 & 0 \\ 0 & -1 & 10 & 0 \\ 0 & 0 & -1 & 10 \end{bmatrix} \text{ and in base } k \text{ would give } B_k = \begin{bmatrix} -1 & k & 0 & 0 \\ 0 & -1 & k & 0 \\ 0 & 0 & -1 & k \end{bmatrix}$$
$$[1,8,5,1] \equiv [1,0,85,1] \equiv [0,18,0,51] \mod \mathcal{L}(B_{10})$$

To decompose vectors in the unique representation we use in “everyday-life”, we would reduce successively the i-th coefficient to the maximum with the i-th vector, from the first to the last. Which, is rightfully so, the equivalent of an euclidean division. Here the reduction works intuitively as we are subtracting some large multiple of k in a position to add a small multiple of 1 in another. What now if we decide to use number systems that are not the “number-system” lattice we showcased? Instead of classical euclidean division we could some form of approximation Babai’s Rounding-Off algorithm (see algorithm 2). Such was the idea of Bajard, Imbert and Plantard [BIP04]: numbers would be represented by vectors, which grows as computations are done but can be reduced by lattice reduction. Thus, the main idea behind **PSW** is there as quoted initially [PSW08]. To know more about lattices used as number systems, we refer to [Pla05] as an entry point. For now, we will continue with the description of the **PSW**-framework.

4.1.2 Spectral Radius and Eigenvalues

While the following mathematical concepts are not needed to understand **DRS**, those are essential to understand the original framework of **PSW**. Those are the exact same definitions given in [PSW08] which itself quotes various books. In all following definitions, $n \in \mathbb{N}$.

Definition 42 (Polytope Norm).
We denote $\|.\|_P$ *as the matrix norm consistent to the vector norm* $\|.\|_P$ *defined as* $\forall v \in \mathbb{C}^n, \|v\|_P = \|vP^{-1}\|_\infty$ *where* P *is invertible.*

To compute the polytope norm $\|.\|_P$ of a matrix, we have $\forall A \in \mathbb{C}^{n,n}, \|A\|_P = \|PAP^{-1}\|_\infty$.

Definition 43 (Eigenvalue).
Let A *be a square matrix in* $\mathbb{C}^{n,n}$*, a complex number* λ *is called a eigenvalue of* A *if there exists a column-vector* $h \neq 0$ *such that* $Ah = \lambda h$*. The column-vector* h *is called an eigenvector of* A*.*

Note that λ is the typical symbol for eigenvalues, but is also the typical symbol for a lattice minima (see definition 13). This is not unusual when we work in-between different fields of mathematics (and/or computer science). While we do use the same symbol here, we will make it clear context-wise when the symbol represents a lattice minima or an eigenvalue. Typically, if we are writing about the convergence of a reduction, the spectral radius or a diagonalization, then we mean an eigenvalue. If we are discussing about the complexity of a lattice problem or the security of a cryptosystem, we mean a lattice minima.

Definition 44 (Spectral Radius).
Let A be a square matrix in $\mathbb{C}^{n,n}$. We denote $\rho(A)$ as the spectral radius of A defined as the maximum of the absolute value of the eigenvalues of A: $\rho(A) = \max\{|\lambda|, Ax = \lambda x\}$.

The spectral radius we just defined is essentially the cornerstone of all analysis provided in [PSW08], which is linked to but not mentioned in the original **DRS** description [PSDS18].

Theorem 5 (Gelfand's spectral radius formula).
$\rho(M) = \lim_{k\to\infty} \|M^k\|^{1/k}$

Gelfand's formula basically states that all norms converges to the spectral radius.

4.1.3 The original PSW framework

While **GGH** and other lattice-based cryptosystems relied on having a "Good" basis as a secret key, the definition of "Good" was dependent often relative to the cryptosystem chosen and an arbitrary intuition. In that sense, [PSW08] gives a specific definition of a good basis.

Definition 45 (A **PSW**-good basis).
*Let D_g, M be two matrices and a lattice $\mathcal{L}$ such that $\mathcal{L} = \mathcal{L}(D_g - M)$. We say $D_g - M$ is **PSW**-good if and only $\rho(MD_g^{-1}) < 1$.*

Note here that D_g does not have to be a diagonal matrix. For efficiency and implementation simplicity however, we usually pick $D_g = D_{Id}$. This definition of a "good" basis is born from an approximation of Babai's Rounding-Off algorithm [Bab86] for **CVP** in maximum norm. With that in mind, we present in Alg 13 the reduction algorithm (which is the signing algorithm) born of this approximated Babai for a lattice $\mathcal{L}$.

However, using a diagonal dominant basis ("weakly" or not), the algorithm can be simplified to what we will call the **PSW**-reduction algorithm (see Algorithm 14).

Algorithm 13 Approximate vector reduction algorithm

```
Require: A vector $v \in$, two matrices $D_g, M$ such that $\mathcal{L} = \mathcal{L}(D_g - M)$ and $D_g$ is diagonal invertible.
Ensure: $w \in \mathbb{Z}^n$ such that $w \equiv v \pmod{\mathcal{L}}$ and $\|w\|_{D_g} < 1$.
$w \leftarrow v$
while $\|w\|_{D_g} \geq 1$ do
    $q \leftarrow \lceil w D_g^{-1} \rfloor$
    $w \leftarrow w - q(D_g - M)$
return $w$
```

Algorithm 14 **PSW** vector reduction algorithm

```
Require: $v \in \mathbb{Z}^n$, $D_g, M \in \mathbb{Z}^{n \times n}$ such that $\mathcal{L} = \mathcal{L}(D_g - M)$ and $D_g$ is diagonal invertible.
Ensure: $w \in \mathbb{Z}^n$ such that $w \equiv v \mod \mathcal{L}$ and $\|w\|_{D_g} < 1$.
$w \leftarrow v$
$i \leftarrow 0$
while $k \geq n$ do
    $k \leftarrow n$
    $q \leftarrow \lfloor w_i / D_{i,i} \rfloor$
    $w_i \leftarrow w_i - q D_{i,i}$
    for $j = 0$ to $n - 1$ do
        $w_{i+j \mod n} \leftarrow w_{i+j \mod n} + q M_{i,j}$
        if $|w_{i+j \mod n}| < D_{i+j \mod n,\ i+j \mod n}$ then $k \leftarrow k + 1$
    $i \leftarrow i + 1$
return $w$
```

A small MAGMA code can be found in the appendix for diagonal dominant lattices(see code A.3). The **PSW** vector reduction algorithm however is not proven to always terminate, and an experimental conjecture was provided to ensure its termination to a solution.

Conjecture 1 (The **PSW** conjecture)**.**
*If $\rho(MD_g^{-1}) < 1/2$, then the **PSW** vector reduction algorithm converges.*

Note that the **PSW** vector reduction algorithm iterates each position successively. It does not have to be the case. Not only there is often more than one valid approximation, but its ordering does not matter much as long as there is no infinite loop: those points can be important for future work in one wishes to pick specific solutions with respect to statistical properties or other conditions.

Example 12. *Example of the reduction with* $v = \begin{bmatrix} 32 & 45 & 37 & 23 \end{bmatrix}$ *and* $D = 10$.

$$M = \begin{bmatrix} 10 & -2 & 3 & 1 \\ 1 & 10 & 3 & 5 \\ 2 & -4 & 10 & 3 \\ -2 & 5 & 2 & 10 \end{bmatrix}$$

$$v \leftarrow v - 3M_1 = \begin{bmatrix} 32 & 45 & 37 & 23 \end{bmatrix} - \begin{bmatrix} 30 & -6 & 9 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 51 & 28 & 20 \end{bmatrix}$$
$$v \leftarrow v - 5M_2 = \begin{bmatrix} 2 & 51 & 28 & 20 \end{bmatrix} - \begin{bmatrix} 5 & 50 & 15 & 25 \end{bmatrix} = \begin{bmatrix} -3 & 1 & 13 & -5 \end{bmatrix}$$
$$v \leftarrow v - 1M_3 = \begin{bmatrix} -3 & 1 & 13 & -5 \end{bmatrix} - \begin{bmatrix} 2 & -4 & 10 & 3 \end{bmatrix} = \begin{bmatrix} -5 & 5 & 3 & -8 \end{bmatrix}$$

Final result:

$$\begin{bmatrix} 32 & 45 & 37 & 23 \end{bmatrix} \equiv \begin{bmatrix} -5 & 5 & 3 & -8 \end{bmatrix} \mod \mathcal{L}(M)$$
$$\begin{bmatrix} 37 & 40 & 34 & 31 \end{bmatrix} \equiv \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \mod \mathcal{L}(M)$$

Check equivalency with the ***HNF***(M)*:*

$$\textit{Start :} \begin{bmatrix} 37 & 40 & 34 & 31 \end{bmatrix}$$

$$\boldsymbol{HNF}(M) = \begin{bmatrix} 7799 & 0 & 0 & 0 \\ 3359 & 1 & 0 & 0 \\ 1053 & 0 & 1 & 0 \\ 3569 & 0 & 0 & 1 \end{bmatrix}$$

4th coefficient:
$$\begin{bmatrix} -110602 & 40 & 34 & 0 \end{bmatrix}$$
3rd coefficient:
$$\begin{bmatrix} -146404 & 40 & 0 & 0 \end{bmatrix}$$
2nd coefficient:
$$\begin{bmatrix} -280764 & 0 & 0 & 0 \end{bmatrix}$$
$$\textit{1st coefficient:} \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

But note how the reduced solution is not unique and

$$\begin{bmatrix} -5 & 5 & 3 & -8 \end{bmatrix} \equiv \begin{bmatrix} 5 & 3 & 6 & -7 \end{bmatrix} \mod \mathcal{L}(M)$$

Given n fixed, the initial first instantation of **PSW** then works as follows:

Setup

- Pick a random matrix $M \in \mathbb{Z}^n$ with "low" values.
- Compute $D = \lfloor 2\rho(M) + 1 \rfloor$
- Compute H be the **HNF** of $\mathcal{L}(D_{Id} - M)$.

The public key is given as (D_{Id}, H) and the secret key M is kept. Note that M was initially set within $\{-1, 0, 1\}^n$ but that was not made mandatory to function, neither was the condition $D = \lfloor 2\rho(M) + 1 \rfloor$.

Sign

Given a message m:

- Hash a message m into a random vector $h(m) = x \in \mathbb{Z}^n$ such that $\|x\|_{D^2_{Id}} < 1$
- Apply the **PSW**-vector reduction into x and save its output w.

The signature is given as w. Note $\|x\|_{D^2_{Id}} < 1$ was also facultative.

Verify

Given a public key (D, H) and a signature w for a message m:

- Check if $\|w\|_{D_{Id}} < 1$.
- Check if $h(w) - m \in \mathcal{L}(H)$.

Checking the second step here is fast given a **HNF** as showed in [Mic01].

Now that we reintroduced the **PSW** signature scheme, note that constructing instances of **PSW** in a fast manner is not trivial: one would need to be able to ensure that the **PSW** conjecture is respected.

Claimed structural security

The main selling point of the **PSW** approach is to be a “cheap” alternative security patch to **GGH**Sign against [NR09, DN12] aside from the one proposed in [GPV08] which was secure but slow.

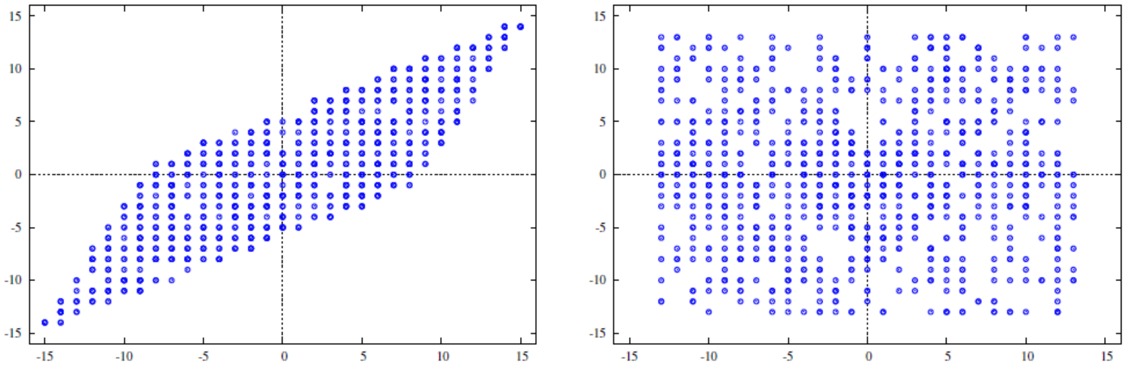


Figure 4.1: Signatures over l_2 and l_∞

The hopes were for the l_∞ norm to be more secure than the l_2 norm, by revealing less structure about the key. Figure 4.1 is taken straight from [PSW08].

4.2 The original DRS scheme

The original definition of the **DRS** scheme can be considered another fork of the **PSW** framework. The lattice admits a diagonal dominant basis, and the signature process uses the **PSW** vector reduction algorithm. Their secret key is a diagonal dominant basis, which is different from the original theoretical **PSW** proposition (although their practical proposition is heuristically a diagonal dominant basis). The coefficient n will denote the dimension unless mentioned otherwise. The initial

DRS scheme requires multiple other parameters to be preset (see the file *api.h* in the NIST submission).

Our unwillingness to use multiprecision arithmetic also restricts **DRS** to use a **HNF** as a public key, and enforces the choice of multiple algorithms and parameters in order to fit every computations within 64-bits. This is mostly due to the licensing and the coding restrictions the NIST enforced for their submissions: without them, the difference between **DRS** and the first proposition for a practical **PSW** would be minimal. We will describe the algorithm and refer to the appendix for a MAGMA implementation. Note that a C implementation of most relevant algorithms should be available on the NIST website [PSDS18].

4.2.1 Setup

Using the same notation as the report given in [PSDS18], we briefly restate all initial algorithms.

Secret key generation

The secret key is a $n \times n$ matrix that contains vectors of equal norm, all generated by an absolute circulant structure. Only 4 coefficients, given publicly, compose each vector: D, B, 1 and 0.

- D, the large diagonal coefficient. This is a basic component in the **PSW**-framework. However, D is fixed equal to n before key generation and not ad-hoc.

- N_B, the number of occurences per vector of the "big" noise $\{-B, B\}$, and is the lowest positive number such that $2^{N_B}\binom{n}{N_b} \geq 2^{\lambda}$. The reasoning behind this parameter is to thwart combinatorial attacks which relies on finding the position of the values B.

- B, the value of the "big" noise, and is equal to $D/(2N_B)$. It is a coefficient that is chosen large to increase the size of the shortest vector in the norm l_2. The purpose of this coefficient was to increase the security of the scheme against pure lattice reduction attacks.

- N_1, the number of values $\{-1, 1\}$ per vector, is equal to $D - (N_B B) - \Delta$. Δ is a constant that will be defined later. The role of those small 1 is to increase the perturbation within each coefficient position per vector when applying the **PSW** vector reduction algorithm.

Those parameters are chosen such that the secret key matrix stays diagonal dominant as per the definition written previously. Algorithm 15 is the original secret key computation. The only difference between the secret key of the first **PSW** instantiation and **DRS** is the noise. As explained in both original works, their estimated security is based on the noise.

Example 13. *Secret key generation.* $D = 6$, $N_B = 2$, $B = 2$, $N_1 = 1$.

Step 1:
$$\begin{bmatrix} \boldsymbol{6} & 2 & 2 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{\text{Random Permutation}} \begin{bmatrix} \boldsymbol{6} & 0 & 2 & 0 & 1 & 2 \end{bmatrix}$$

Step 2:
$$\begin{bmatrix} \boldsymbol{6} & 0 & 2 & 0 & 1 & 2 \end{bmatrix} \xrightarrow{\text{Circulant Matrix}} \begin{bmatrix} \boldsymbol{6} & 0 & 2 & 0 & 1 & 2 \\ 2 & \boldsymbol{6} & 0 & 2 & 0 & 1 \\ 1 & 2 & \boldsymbol{6} & 0 & 2 & 0 \\ 0 & 1 & 2 & \boldsymbol{6} & 0 & 2 \\ 2 & 0 & 1 & 2 & \boldsymbol{6} & 0 \\ 0 & 2 & 0 & 1 & 2 & \boldsymbol{6} \end{bmatrix}$$

Step 3:
$$\begin{bmatrix} \boldsymbol{6} & 0 & 2 & 0 & 1 & 2 \\ 2 & \boldsymbol{6} & 0 & 2 & 0 & 1 \\ 1 & 2 & \boldsymbol{6} & 0 & 2 & 0 \\ 0 & 1 & 2 & \boldsymbol{6} & 0 & 2 \\ 2 & 0 & 1 & 2 & \boldsymbol{6} & 0 \\ 0 & 2 & 0 & 1 & 2 & \boldsymbol{6} \end{bmatrix} \xrightarrow{\text{Random Signs}} \begin{bmatrix} \boldsymbol{6} & 0 & -2 & 0 & -1 & 2 \\ 2 & \boldsymbol{6} & 0 & -2 & 0 & -1 \\ 1 & -2 & \boldsymbol{6} & 0 & 2 & 0 \\ 0 & -1 & 2 & \boldsymbol{6} & 0 & 2 \\ -2 & 0 & 1 & -2 & \boldsymbol{6} & 0 \\ 0 & -2 & 0 & 1 & 2 & \boldsymbol{6} \end{bmatrix}$$

Algorithm 15 Secret key generation

```
Require: A random seed $x$
Ensure: A secret key $x, S = D_{Id} - M$
$S \leftarrow 0$
$t \in \mathbb{Z}^n$
$InitiateRdmSeed(x)$                                   ▷ Sets the randomness via $x$
$t \leftarrow [D,\underbrace{B,...,B}_{N_B},\underbrace{1,...,1}_{N_1},0,...,0]$     ▷ Sets initial rotating vector
$t \leftarrow$ RdmPmtn$(t)$                              ▷ Shuffle non-$D$ positions randomly
for $i = 1$ ; $i \le n$ ; $i = i + 1$ do
    $S[i][i] \leftarrow t[1]$                            ▷ Set diagonals coefficient $D$
    for $j = 2$ ; $j \le n$ ; $j = j + 1$ do
        $c \leftarrow t[j] *$ RdnSgn()                   ▷ Set others with random signs
        $S[i][((i + j) \mod n) + 1] \leftarrow c$
return $x, S$
```

Public key generation

The lattice of the public key P_k is the same lattice as the secret key S_k. However, we provide a different basis, which is more in tune with a classical **GGH** approach of "good" and "bad" basis. Roughly speaking, we need to provide an unimodular transformation matrix T such that $P_k = TS_k$. We have three objectives:

- Construct T in a fast manner, from a large combinatorial set.
- Bound the coefficients of P_k, making sure computations do not overflow.
- Make sure T^{-1} is hard to reconstruct.

The third objective will rely on assumptions, as we cannot prove it at this point for any T (except specific unique forms like the **HNF**). The first two objectives, however, are reasonably achievable. First of all, we can easily to include permutation matrices to construct T: They respect the first two objectives. However, in the case of diagonal matrices, it is easy to see the third point is discarded with just permutations: A diagonal dominant structure is easy to "permute" back. The problem then will be to intermingle row vectors and control their growth without changing the lattice generated. We here choose the intermingling of 2 vectors to be equivalent to a multiplication of random pairs of vectors (a $2 \times n$ matrix) by a square unimodular matrix of dimension 2 and maximum norm of 2.

The set $U_{\{+,-\}}$ of the unimodular matrices we use for the purpose of intermingling vectors is very particular:

$$U_{\{+,-\}} = \left\{ U_+ = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, U_- = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \right\}$$

and let us define the set $U'_{\{+,-\}}$ constructed from $U_{\{+,-\}}$:

$$U'_{\{+,-\}} = \left\{ \forall i \in [1, n/2],\ U_i \in U_{\{+,-\}} : \begin{bmatrix} U_0 & 0 & \dots & & 0 \\ 0 & U_1 & \ddots & & \\ \vdots & 0 & \ddots & \ddots & \vdots \\ & & \ddots & U_{n/2-1} & 0 \\ 0 & \dots & & 0 & U_{n/2} \end{bmatrix} \right\}$$

Let $P \in S_n$ a permutation matrix and $U \in U'_{\{+,-\}}$, and M a structured matrix we want to make hard to recover. We can conceive a "round" of scrambling to be the transformation $M \longleftarrow UPM$. In our case one single round of scrambling is obviously not enough. Therefore, we need to scramble multiple times, each new round being applied with a new randomly selected tuple (U, P). Let R be the number of such rounds. Our choice for T such that $P_k = TS_k$ is thus:

$$U = P_{R+1} \prod_{i=1}^{R} U_i P_i$$

i.e., a combination of $R+1$ permutations and R intermingling of vectors.

The number of rounds R is decided upon security consideration but also efficiency reasons as we wanted to fit every computation within 64-bits. Each round multiplies the maximum size of the coefficients (we will denote δ) by a factor at most 3. Note that the case 3 is rare. The number R is dependent of other parameters we will explain later.

The public key is thus by successive additions/substractions of pair of vectors (see Algorithm 16). Note that the only difference with the original scheme [PSDS18] is that we do not store the $\log_2$ of the maximum norm. We estimate this information to be easily computed at negligeable time. A MAGMA code can be found in the appendix (see code Figure A.4).

Algorithm 16 Public key generation

```
Require: $S = D_{Id} - M$ the reduction matrix, a random seed $x$
Ensure: $P$ such that $\mathcal{L}(P) = \mathcal{L}(S)$ and $\|S\|_\infty << \|P\|_\infty \leq 3^R \|S\|_\infty$
P ← S
InitiateRdmSeed(x)                              ▷ Sets the randomness via x
for $i = 1$ ; $i < R$ ; $i = i + 1$ do
    P ← RdmPmtn(P)                              ▷ Shuffle the rows of P
    for $j = 1$ ; $j \leq n - 1$ ; $j = j + 2$ do
        t ← RdmSgn()
        $P[j] = P[j] + t * P[j+1]$              ▷ "Random" linear combinations
        $P[j+1] = P[j+1] + t * P[j]$
P ← RdmPmtn(P)
return P
```

The power of 2 p_2 we removed from the descrition has no security impact, and is used mostly for the verification process to make sure intermediate computation results stay within 64-bits. This type of public key is very different from the **HNF** [PSW08] suggested to use, however the computation time of a **HNF** is non-negligible. As we will see later this directly impact the signature.

4.2.2 Signature

Rather than checking if the successive approximation of Babai's algorithm on a vector m of converges [PSW08], **DRS** checks if the successive approximation on a vector m can reach a point where $\|m\|_1 < nD$, and if $\exists i, |m_i| > D$, reduce m further without increasing $\|m\|_1$.

Given the fact that the secret key is a diagonally dominant matrix, Alg 17 is guaranteed to complete: forcing $tr(M) = 0$ on the noise, we presented a proof that ignored the convergence of the reduction steps but showed the existence of a reachable valid solution for $\|m\|_\infty < D$. A MAGMA code of the signing algorithm can be found in the appendix (see the code Figure A.5). In a certain sense, it uses the fact that the **PSW** vector reduction algorithm (Algorithm 14) does not need to converge to find a solution. The original proof can be seen in [PSDS18]; however, we are not going to mention it here since a better proof will be shown after modification.

Algorithm 17 Sign: Coefficient reduction first, validity vector then

```
Require: $v \in \mathbb{Z}^n$, $(x, S)$ the secret seed and diagonal dominant matrix
Ensure: $w$ with $v \equiv w\ [\mathcal{L}(S)]$, $\|w\|_\infty < D$ and $k$ with $kP = v - w$
$w \leftarrow v$, $i \leftarrow 0$, $k \leftarrow [0, ..., 0]$
while $\|w\|_\infty < D$ do                          ▷ Apply the PSW vector reduction
    $q \leftarrow w_i/D$
    $k_i \leftarrow k_i + q$                                      ▷ Ensure $kS = v - w$
    $w \leftarrow w - qS[i]$
    $i \leftarrow i + 1 \bmod n$
$InitiateRdmSeed(x)$                                 ▷ Set randomness identical to Setup
for $i = 1$ ; $i \leq R$ ; $i = i + 1$ do                 ▷ Transform $kS = v - w$ into $kP = v - w$
    $k \leftarrow$ RdmPmtn($k$)
    for $j = 1$ ; $j \leq n - 1$ ; $j = j + 2$ do
        $t \leftarrow$ RdmSgn()
        $k[j + 1] = k[j + 1] - t * k[j]$
        $k[j] = k[j] - t * k[j + 1]$
$k \leftarrow$ RdmPmtn($k$)
return $k, v, w$
```

Another difference with the original **PSW** is the fact that it did not have a second vector k to output in their initial scheme and thus only had to deal with the reduction part [PSW08]. The vector k is needed to ensure $v - w \in \mathcal{L}(P_k)$, which in the case of a **HNF** was not needed as the triangular form allowed an easy verification.

Note that if we wish to fit every computation within 64-bits, then we need to enforce $\log_2 \|k\| < 63$. Thus we need to bound it with previous parameters, i.e.,

$$k'(D - M) = v - w$$

$$\|k'\| \leq \|v - w\| \|(D - M)^{-1}\|$$

$$\|k'\| \leq \|v - w\| \|D^{-1} \tfrac{1}{1-\frac{M}{D}}\|$$

$$\|k'\| \leq \|v - w\| \|D^{-1}\| \| \tfrac{1}{1-\frac{M}{D}}\|$$

$$\|k'\| \leq \|v - w\| \|D^{-1}\| \| \|1 + \tfrac{M}{D} + (\tfrac{M}{D})^2 + ...\|$$

$$\|k'\| \leq \|v - w\| \|D^{-1}\| (\|1\| + \|\tfrac{M}{D}\| + \|\tfrac{M}{D}\|^2 + ...)$$

$$\|k'\| \leq \|v - w\| \|D^{-1}\| \| \tfrac{1}{1-\|\frac{M}{D}\|}\|$$

$$\|k'\| \leq \|v - w\| \| \tfrac{1}{D-\|M\|}\|$$

$$\|k'\| \leq \|v - w\| \tfrac{1}{\Delta}$$

$$\|k'\| \leq (\delta + 1)\tfrac{1}{\Delta} = \tfrac{\delta+1}{\Delta}$$

therefore:

$$\begin{aligned}
k &= k'U^{-1} \\
\|k\| &\leq \|k'\|\|U^{-1}\| \\
\|k\| &\leq \|\frac{\delta+1}{\Delta}\|\|U^{-1}\| \\
\|k\| &\leq \frac{(\delta+1)3^R}{\Delta}
\end{aligned}$$

thus giving us the means to fix Δ, δ, R to fit every coefficients within 64-bits.

4.2.3 Verification

Given a hashed message vector v, the signature (k, w), the verification is reduced to the equality test $kP_k = (v - w)$. However, as the computation kP_k might overflow (the maximum size of k depends of δ, Δ, R, and P_k's ones from D, R). In the following verification algorithm we recursively cut k into two parts $k = r + p_2 q$ where p_2 is a power of 2 that is lower than $2^{63}/\|P_k\|$, which ensures rP_k is not overflowing.

Given $P_k, 2^k$, $t = v - w$ and $k = r + p_2 q$ with $\|r\| < p_2$, we have $kP_k - t = c$ with $c = 0$ if and only if $kP_k = v - w$. Therefore

$$qp_2 P_k + rP_k - t = c \quad \rightarrow \quad qP_k = \tfrac{c+t-rP_k}{p_2}$$

and thus p_2 should divide $t-rP_k$ if $c = 0$: if not, that means $c \neq 0$ and the verification returns FALSE. Otherwise, we set $k' \leftarrow q$ and $t' \leftarrow t - rP_k$ and repeat

$$(qP_k - \tfrac{t-rP_k}{p_2} = \tfrac{c}{p_2}) \rightarrow (k'P_k - t' = c')$$

where c' becomes exactly the integer c/p_2 regardless of its value (if it didn't fail before). The verification stops when both $t' = 0$ and $k' = 0$. Note that both need to be 0 at the same time, if only one of them is 0 then the verification fails.

The verification, given k, v, w, P_k is then as follow in algorithm 18. Note that the core algorithm could be optimized but we just give here the overall idea. A MAGMA code is provided in the appendix (see code Figure A.7) for testing purposes.

Algorithm 18 Verify

```
Require: $v, w, k \in \mathbb{Z}^n$, $P$ the public key
Ensure: Checks $v \equiv w \ [\mathcal{L}(P)]$ and $\|w\|_\infty < D$
if $\|w\|_\infty >= D$ then                        ▷ Checks $\|w\|_\infty < D$
    return FALSE
$q \leftarrow k$
$t \leftarrow v - w$
$p_2 \leftarrow \log_2 \|P\|_\infty$
while $q \neq 0 \wedge t \neq 0$ do                ▷ Verification per block of size $p_2$
    $r \leftarrow q - (p_2 \times \lceil q/p_2 \rfloor)$        ▷ Get the smallest sized remainder
    $t \leftarrow t - (r * P)$
    if $t \neq 0 \mod p_2$ then                    ▷ Check block
        return FALSE
    $t \leftarrow t/p_2$                           ▷ Update values for next iteration
    $q \leftarrow (q - r)/p_2$
    if $(t = 0) \veebar (q = 0)$ then
        return FALSE
return TRUE
```

If multiprecision integers were to be used (as using GNU Multiple Precision Arithmetic Library (GMP)), it would not take a while loop with multiple rounds to check. Whether this is more efficient or not remains to be tested.

Example 14. *Verification example for $p_2 = 10000$:*

$$P = \begin{bmatrix} -1840 & 2471 & -382 & -820 & 710 & 3048 \\ 1966 & -1378 & 1486 & 1721 & 1430 & -4090 \\ -1998 & 4317 & 994 & 271 & 3660 & 2211 \\ 2729 & -3460 & 746 & 1375 & -680 & -4662 \\ 2784 & -6566 & -1866 & -801 & -6100 & -2700 \\ 3679 & -3323 & 2144 & 2716 & 1380 & -7160 \end{bmatrix}$$

$$k = \begin{bmatrix} -54029 & -77227 & 6908 & -38654 & -4594 & 50148 \end{bmatrix}$$

$$v = \begin{bmatrix} 924 & 232 & 131 & 692 & 439 & 694 \end{bmatrix}$$

$$w = \begin{bmatrix} 0 & 9 & -9 & -1 & -1 & 0 \end{bmatrix}$$

Goal: verify $kP = v - w = \begin{bmatrix} 924 & 223 & 140 & 693 & 440 & 694 \end{bmatrix}$ with low size computations.
Set $q = k$ and $t = v - w$.

First pass:

$$r \leftarrow q \mod p_2 = \begin{bmatrix} -4029 & 2773 & -3092 & 1346 & -4594 & 148 \end{bmatrix}$$

$$t \leftarrow t - r \times P =$$

$$\begin{bmatrix} -10470000 & 2110000 & -12480000 & -13170000 & -17100000 & 25390000 \end{bmatrix}$$

t is clearly divisible by p_2, update q, t

$$q \leftarrow (q - r)/p_2 = \begin{bmatrix} -5 & -8 & 1 & -4 & 0 & 5 \end{bmatrix}$$

$$t \leftarrow t/p_2 = \begin{bmatrix} -1047 & 211 & -1248 & -1317 & -1710 & 2539 \end{bmatrix}$$

Both are non-zero. Repeat.

Second pass:

$$r \leftarrow k \mod p_2 = \begin{bmatrix} -5 & -8 & 1 & -4 & 0 & 5 \end{bmatrix}$$

$$t \leftarrow t - r \times P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

t is clearly divisible by p_2, update q, t and continue

$$q \leftarrow (q - r)/p_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$t \leftarrow t/p_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Both are zero. End with true.

4.3 On the Security of the Public Key

Note that the public key of **DRS** relies on successive multiplication of heavily structured 2×2 matrices. There is no concrete security reduction or previous examples in the literature to assert the security of this type of public key. However, the main objective of the public key setup of **DRS** was to "evenly distribute" the coefficients around all positions while ensuring the setup could never overflow (on 64-bits processors). If this specific method ever finds a weakness, we could either use a **HNF** which can be computed in polynomial time [PS10], or use other types of unimodular matrices. **GGH** for example used triangular matrices to generate their keys. Other methods of sampling are welcomed; however, to the best of our knowledge the **HNF** still provides optimal safety as it is unique per lattice and an attack on the structure of the **HNF** is therefore, an attack on all possible basis [Mic01].

The problem with a **HNF** is its computation time and the objects we need to manipulate: Multiprecision library are often needed and computation time for cryptographically secure sizes goes well over a dozen of seconds even on high-end computers, which is a severe flaw for a lot of applications. While speeding up the computations for this particular type of keys might be possible, it was; however, not the point of our work so far. We here focus on patching the structure of the secret key, as this is the only angle where flaws were discovered in the literature.

4.3.1 Li, Liu, Nitaj and Pan's Attack

In ACISP 2018, Li, Liu, Nitaj and Pan [LLNP18] presented an attack that makes use of short signatures to recover the secret key. Their observation is that two different signatures from the same message is also a short vector of the lattice. Then, gathering sufficient number of short vectors enable easier recovery of the secret key using lattice reduction algorithms with the vectors generated. Their suggestion to fix this issue is to either store previous signed messages to avoid having different signatures, or padding a random noise in the hash function. We should note that the initial **DRS** scheme is not randomized as the algorithm is deterministic and produce a unique signature per vector.

We do note that the authors of **DRS** suggested in their report [PSDS18] to use a random permutation to decide the order of the coefficent reduction, and thus Li, Liu, Nitaj and Pan's attack might apply to their suggestion. However, the order of the coefficient reduction could also be decided deterministically by the hashed message itself, and therefore, Li, Liu, Nitaj and Pan's attack is not fully applicable, as this method would produce an unique signature per message. They can still generate a set of relatively short vectors $(r_1, \ldots, r_2) \in \mathcal{L}^n$ of the lattice $\mathcal{L}$; however, it is unclear whether the specialized version of their attack using vectors s,$(v_1, \ldots, v_n)$ where $s - v_i \in \mathcal{L}$ is still applicable. It seems to be easier to recover the key when using multiple signatures from the same message as a lattice basis when using lattice reduction algorithms rather than using random small vectors of the lattice: This could imply that diagonal dominant basis have inner weaknesses beyond the simple instantiation of **DRS**. From our understanding, the secret key matrices they generated for their tests used a noise matrix $M \in \{-1, 0, 1\}^{n \times n}$, which could have had an impact in their experimentations. It is still unknown if other noise types such as the ones in **DRS** or the type of noise we are about to propose are affected: To the best of our knowledge, **DRS** was not quoted in their work.

We stress that we do not claim the new setup to be perfectly secure against Li, Liu, Nitaj and Pan's attack, we merely claim more experimentations would need to be done as of now. Furthermore, the countermeasures proposed by Li, Liu, Nitaj and Pan also apply to those new keys, and should be applied if one wishes for a more concrete security. The next attack, however, does not have clear known countermeasures as of now and is the main focus of this paper.

4.3.2 Yu and Ducas's Attack

We explained in the previous section about the security of **DRS** against Li, Liu, Nitaj and Pan's attack. On the other hand, it is unclear if such a modification would add an extra weakness against Yu and Ducas's heuristic attack. Their attack work in two steps. The first one is based on recovering certain coefficients of a secret key vector using machine learning and statistical analysis. The second is classical lattice-reduction attack to recover the rest of the secret key.

For the first step, Yu and Ducas noticed that the coefficients $\boldsymbol{B}$ of the secret key and the 1 could be distinguished via machine learning techniques [YD18a], noticing for one part that the non-diagonal coefficients follow an "absolute-circulant" structure, and the fact that only two types of non-zero values exist. Based on this information, a surprisingly small amount of selected "features" to specialize a "least-square fit" method allowed them to recover both positions and signs of all if not most coefficients $\boldsymbol{B}$ of a secret vector. We note they did not conduct a exhaustive search on all possible methods according to their paper thus stressing that their method might not be the best. We did not conduct much research on the related machine learning techniques; therefore, we cannot comment much on this part as of now.

A few points were presented to explain why their technique works. One point is the difference between the noise coefficients: It was either close to non-existant or very large, causing wave-shaped reductions that could be detected given enough samples. The other point is that this wave-shaped reduction is absolute-circulant, which makes the structure more obvious as this wave-shaped perturbation translates in incremental order. Figure 4.2 is a visual representation of the cascading phenomenon, taken directly from [YD18a] (S is a secret key vector and w a vector to reduce).

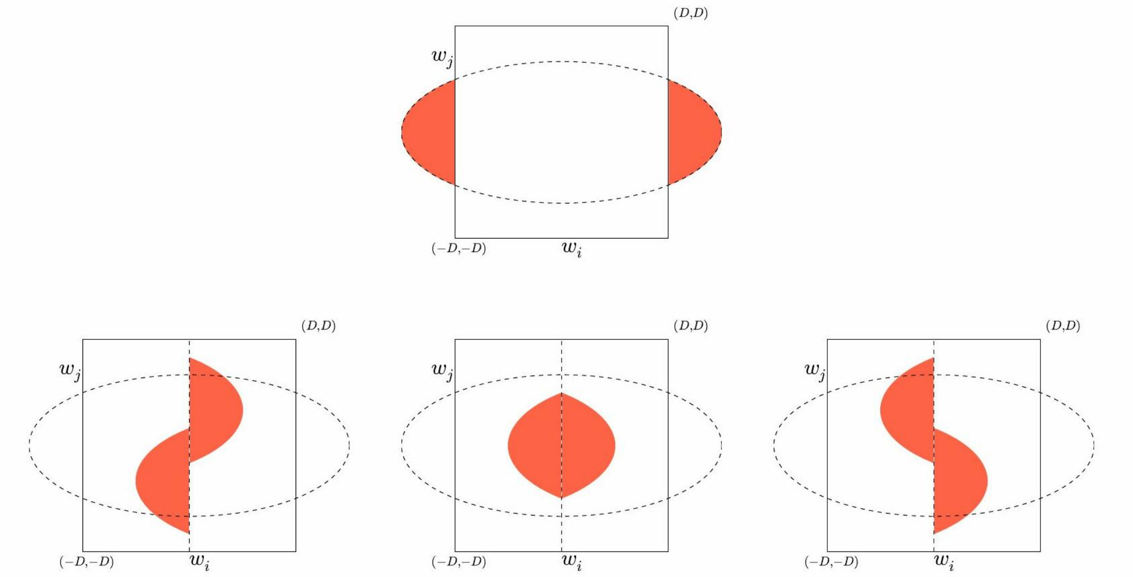


Figure 4.2: Figures in the second row show the regions to which (w_i, w_j) in two cap regions will be moved by reduction at index i when $S_{i,j} = -b, 0, b$, respectively, from left to right.

On the second step, the recovered coefficients and their positions and signs allowed them to apply the Kannan embedding attack on a lattice with the exact same volume as the original public key but of a much lower dimension than the original authors of **DRS** based their security on, by scrapping the known B noise coefficients. Strictly speaking, using the same notation as in the previous description of DRS and assuming the diagonal coefficient is equal to the dimension, the initial search of a shortest vector of length $\sqrt{B^2N_b + N_1 + 1}$ in a lattice of dimension n of determinant n^n becomes a search of a shortest vector of length $\sqrt{N_1 + 1}$ in a lattice of dimension $n - N_b$ of determinant n^n. A visual representation on the effect of this attack can be seen in the next section or in Example 13 where all big red coefficients are replaced by 0 in one basis vector. The efficiency of lattice reduction techniques then affects the evaluation of the security strength of the original **DRS** scheme.

Yu and Ducas conducted experiments and validated their claims using only a few dozens of thousands of signatures per key, reducing the security of the initial submission of **DRS** from 128-bits to maybe at most 80-bits, using BKZ-138. The original concept (not the instantiation) from [PSW08], however, still seems to be safe for now: While it has no security proof, to the best of our knowledge, no severe weaknesses have been found so far. Furthermore, Yu and Ducas advised of some potential countermeasures to fix **DRS**, i.e., breaking the structure of the particular instance that was submitted: The deterministic approach of the number of B, 1, being limited to those two values (5 if we consider zeroes and signs), and the “absolute-circulant” structure. They also pointed that a lack of security proof could be problematic and gave some opinions about how one can potentially find provable security for the **DRS** scheme.

We invite readers to read their work: It is possible that new techniques relying on machine learning could apply to all lattice-based cryptosystems beyond **DRS** by tweaking their process for each specific structure.

In the following section, we provide a countermeasure which follows some of the recommendations given by Yu and Ducas as breaking the secret key noise structure and giving some statistical heuristic, while still preserving the original idea given in PKC 2008 [PSW08].

4.4 New Setup

We do not change any algorithm here aside the setup of the secret key: the public key generation method is left unchanged, along with the signature and verification. Compared to the old scheme, this new version is now determined by less parameters, which leave 6 of them using the previous **DRS**: the dimension n, a random generator seed s, a signature bound D, a max norm for hashed messages δ, a sparsity parameter Δ that we always set to one, and R a security parameter determining the number of multiplication rounds to generate the public key.

We choose random noise among all the possible noises vectors which would still respect the diagonal dominant property of the secret key. This choice is following Yu and Ducas's suggestions on breaking the set of secret coefficients, the "absolute-circulant" structure of the secret key, and allowing us to provide statistical evidence. Roughly speaking, we aimed to transform the following structure of

$$\begin{bmatrix} \mathbf{15} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{15} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{15} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{15} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{15} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{15} \end{bmatrix} + \xcancel{\begin{bmatrix} 0 & 5 & 1 & 0 & -1 & 1 \\ -1 & 0 & -5 & 1 & 0 & -1 \\ -1 & 1 & 0 & 5 & 1 & 0 \\ 0 & 1 & 1 & 0 & 5 & -1 \\ 1 & 0 & -1 & 1 & 0 & -5 \\ 5 & -1 & 0 & 1 & -1 & 0 \end{bmatrix}}$$

to something "less-structured", more "random" but still diagonal dominant like

$$\begin{bmatrix} \mathbf{15} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{15} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{15} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{15} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{15} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{15} \end{bmatrix} + \left[\begin{array}{cccccc|c} 4 & -2 & 0 & 3 & -1 & 4 & = 14 \\ -2 & 3 & -1 & 0 & -8 & 0 & = 14 \\ 6 & 1 & 2 & -1 & 1 & 3 & = 14 \\ 0 & 0 & -4 & 3 & 2 & 3 & = 12 \\ -3 & 2 & -1 & -3 & -1 & 3 & = 13 \\ 1 & -1 & 2 & -4 & -4 & 2 & = 14 \end{array}\right]$$

While we want to have random noise, we must ensure we can still sign every message and thus guarantee the diagonal dominant structure of our secret key. Hence, the set of noise vectors we need to keep are all the vectors $v \in \mathbb{Z}^n$ that have a taxicab norm of $\|v\|_1 \leq D - 1$. Let us call that set V_n.

Sampling from V_n, however, is no trivial task. However, preceding work in the academic literature allows us to:

1. Count all points of $\mathbb{Z}^n$ inside a n-ball for the l_1-norm, i.e., $|V_n|$. [SS00]

2. Know how many of them have a fixed amount of zeroes. [SS00]

3. Sample uniformly from the n-simplex, fixing a certain amount of zeroes. [ST04]

Therefore, the plan is the following:

1. Creates a cumulative frequency distribution table from [SS00].

2. Use the table to sample uniformly a number of zeroes.

3. Sampling uniformly within the n-ball of with a fixed number of zeroes.

This new setup will also change the bounds used for the public key, as the original **DRS** authors linked several parameters together to ensure computations stay within 64 bits. However, our paper has a more theoretical approach and we do not focus on the technical implementations.

4.4.1 Picking the Random vectors

We are aiming to build the new noise matrix M, which is a $n \times n$ matrix such that $M \in V_n^n$. In that regard, we construct a table we will call T with D entries such that

$$T[i] = \#\text{vectors } v \in V_n \text{ with } i \text{ zeroes.}$$

This table is relatively easy to build and does not take much time, one can for example use the formulas derivated from [SS00, KGP+89].

From this table, we construct another table T_S such that $T_S[k] = \sum_{i=0}^{k} T[i]$.

The generation algorithm of the table T_S, which we will use as a precomputation for our new setup algorithm can be seen in Algorithm 19.

Algorithm 19 Secret key table precomputation

```
Require: all initial parameters
Ensure: $T_S$ the table sum
$m \leftarrow \mathbf{min}(n, D)$
$T \leftarrow \{1\}^{m+1}$
$T_S \leftarrow \{1\}^{m+1}$
for $j = 2$ ; $j \leq D$ ; $j = j + 1$ do                    ▷ Loop over the norm
    for $i = 2$ ; $i \leq m + 1$ ; $i = i + 1$ do            ▷ Loop over possible non-zeroes
        $x \leftarrow 2^{i-1}\binom{n}{i-1}\binom{j-1}{i-2}$
        $T[m + 1 - i] \leftarrow T[m + 1 - i] + x$
for $i = 1$ ; $i \leq m$ ; $i = i + 1$ do                    ▷ Construct array $T_S$ from $T$
    $T[i + 1] \leftarrow T[i + 1] + T[i]$
$T_S \leftarrow T$
return $T_S$
```

Let us denote the function $Z(x) \rightarrow y$ such that $T_S[y-1] < x \leq T_S[y]$. Since T_S is trivially sorted in increasing order $Z(x)$ is nothing more than a dichotomy search inside an ordered table. If we pick randomly x from $[0; T_S[D-1]]$ from a generator with uniform distribution $g() \rightarrow x$ then we got $Zero() \rightarrow Z(g(x))$ a function that selects uniformly an amount of zeroes amount all vectors of the set V_n, i.e.,

$$Zero() \rightarrow \#\text{zeroes in a random } v \in V_n$$

Now that we can generate uniformly the number of zeroes we have to determine the coefficients of the non-zero values randomly, while making sure the final noise vector is still part of V_n. A method to give such a vector with chosen taxicab norm is given in [ST04] as a correction of the Kraemer algorithm. As we do not want to choose the taxicab norm M directly but rather wants to have any random norm available, we add a slight modification: The method in [ST04] takes k non-zero elements $x_1, \ldots, x_k$ such that $x_i \leq x_{i+1}$ and forces the last coefficient to be equal to the taxicab norm chosen, i.e., $x_k = M$. By removing the restriction and using

$x_k \le D$, giving the amount of non-zero values, we modify the method to be able to take over any vector values in V_n with the help of a function we will call

$$\textbf{KraemerBis}(z) \to \text{random } v \in V_n$$

such that v has z zeroes which is described in Algorithm 20

Algorithm 20 KraemerBis

```
Require: all initial parameters and a number of zeroes $z$
Ensure: a vector $v$ with $z$ zeroes and a random norm inferior or equal to $D$
$v \in \mathbb{N}^n$
$0 \le x_0 < x_1 < \ldots < x_{n-z} \le D$                ▷ Pick randomly $n - z + 1$ elements
for $i = 1$ ; $i \le n - z$ ; $i = i + 1$ do
    $v[i] \leftarrow x_i - x_{i-1}$
for $i = n - z + 1$ ; $i \le n$ ; $i = i + 1$ do
    $v[i] \leftarrow 0$
return $v$
```

With both those new parts, the new setup algorithm we construct is presented in Algorithm 21 using Kraemer bis. We note that in our algorithm, the diagonal coefficient in the secret key is not guaranteed to be equal to the bound used for the maximum norm of the signatures. Nevertheless, we will show that the termination is still ensured in Section 4.4.2. This heavy setup naturally affects the speed of the **DRS** setup, as we noticed in our experiments as shown in Section 4.4.5.

Algorithm 21 New secret key generation

```
Ensure: all initial parameters and another extra random seed $x$
Require: $x, S$ the secret key
$S \leftarrow D_{Id}$
$t \in \mathbb{Z}^n$
$InitiateRdmSeed(x)$                                  ▷ Set randomness
for $i = 1$ ; $i \le n$ ; $i = i + 1$ do
    $Z \leftarrow Zero()$                             ▷ Get the number of zeroes
    $t \leftarrow \textbf{KraemerBis}(Z)$
    for $j = 1$ ; $j \le n - Z$ ; $j = j + 1$ do      ▷ Randomly switch signs
        $t[j] \leftarrow t[j] \times \textbf{RdmSgn}()$
    $t \leftarrow \textbf{RdmPmtn}(t)$                ▷ Permutes everything
    $S[i] \leftarrow S[i] + t$
return $x, S$
```

4.4.2 A Slightly More General Termination Proof

The proof stated in the **DRS** report on the NIST website [PSDS18] was considering that the diagonal coefficient of $S = D_{Id} + M$ stayed equal to the signature bound (i.e., $tr(M) = 0$), which is not this case. We show here that the reduction is still guaranteed nevertheless. Suppose that some coefficients of the noise matrix M are non-zero on the diagonal. Re-using for the most part notations of the original report, where:

- m is the message we want to reduce, which we update step by step
- M is the noise matrix (so M_i is the i-th noise row vector).
- D is the signature bound for which the condition $\|m\|_\infty < D$ has to be verified. We note d_i the i-th diagonal coefficient of the secret key S.

Obviously, the matrix will still be diagonal dominant in any case. Let us denote d_i the diagonal coefficient $S_{i,i}$ of $S = D_{Id} - M$.

If $D > d_i$ we can use the previous reasoning and reduce $\|m_i\|_1$ to $\|m_i\|_1 < d_i < D$, but keep in mind we stop the reduction at $\|m_i\|_1 < D$ to ensure we do not leak information about the noise distribution.

Now $d_i > D$ for some i: reducing to $|m_i| < d_i$ is guaranteed but not sufficient anymore as we can reach $d < |m_i| < d_i \leq D + \Delta < 2d$. Let us remind that $\Delta = D - \sum_{j=1}^{n} |M_{i,j}|$, where Δ is strictly positive as an initial condition of the **DRS** signature scheme (both on the original submission and this paper), $d_i = D + c$ where $c = |M_{i,i}|$.

Without loss of generality as we can flip signs, let us set $m_i = D + k < d_i = D + c$ with $k \geq 0$ the coefficient to reduce. Substracting by S_i transforms

$$m_i \longleftarrow (D + k) - d_i = (D + k) - (D + c) = k - c < 0$$

with $D > c > k \geq 0$. Therefore the reduction of $\|m\|_1$ without the noise is

$$\|m\|_1 \longleftarrow \|m\|_1 - (D + k) + (c - k) = \|m\|_1 - (D - c) - 2k.$$

but the noise contribution on other coefficients is at worst $(D - \Delta) - c$ thus

$$\|m\|_1 \longleftarrow \|m\|_1 - (D - c) - 2k + (D - c - \Delta). \ \|m\|_1 \longleftarrow \|m\|_1 - 2k - \Delta = \|m\|_1 - (2k + \Delta).$$

where $2k + \Delta > 0$. Therefore the reduction is also ensured in the case $d_i > D$.

4.4.3 On Exploiting the Reduction Capacity for Further Security

Remark that the proof hints at the fact we can actually lower the norm $\|m\|_1$ of some vector m to some value lower than D. It is easy to see that when $M = 0$ and $S = D_{Id}$, every coefficient of m can be reduced to $\|m\|_1 < D/2$ in exactly n iterations of the **PSW** vector reduction algorithm. Clearly, there should be some gap between the bound D and the amount of noise in M that can be filled. If we do fill that gap, we can extend the number of available keys to use by extending the set of applicable noise and hopefully making cryptanalysis harder. While it is hard to find examples in a "printable" size where the **PSW** Conjecture (Conjecture 1 in Section 4.1.2) applies while the **DRS** reduction proof does not, it becomes easier as the dimension grows. Using the code in Figure A.8 gives us an example on the gap between the **PSW** conjecture and the **DRS** proof. The output is shown in Figure 4.3

```
Random Seed is 1515430315
Diagonal Value D is 51
Dimension N is 51

Spectral Radius
0.491115563770558861830554360652
Minimum/Maximum l1 norm of noise vectors
59 94
Average l1 norm of noise vectors
76
```

Figure 4.3: Example output where the DRS bound fails but the PSW bound passes

We can see in Figure 4.3 that every noise vector comfortably goes over the **DRS** bound (here $D = 51$) while $\rho(MD^{-1}) \approx 0.49 < 0.5$. Note that the opposite is also true: By changing the noise to enforce the respect of the **DRS** bound (commenting line 14 and uncommenting line 16 of code in Figure A.8), we can obtain the inverted result as seen in Figure 4.4.

```
Random Seed is 1515430315
Diagonal Value D is 51
Dimension N is 51

Spectral Radius
0.510708190604545795839616917492
Minimum/Maximum l1 norm of noise vectors
17 32
Average l1 norm of noise vectors
25
```

Figure 4.4: Example output where the PSW bound fails but the DRS bound passes

One part of an explanation to this phenomenon is that the sign does not affect the **DRS** bound while it does heavily affect the **PSW** bound. If weakness appears on this new **DRS** instantiation due to the noise being too low, intuitively we think increasing the bound of the n-dimensional ball from which we uniformly sample the noise should still lead to most keys being usable w.r.t the **PSW**-conjecture. However, we discuss in the following part methods to efficiently generate keys for **PSW** that are proven to respect the **PSW** conjecture. While they do seem to be relatively simple, establishing instances of a general **PSW** scheme beyond a noise $M \in \{-1, 0, 1\}^{n,n}$ seems to have been lacking in the literature. We hope this can help close the gap between the conclusions of **DRS** and **PSW**.

4.4.4 Ensuring the termination of PSW

In this subsection we present simple ways for the **PSW** approach to be more practical. A first example can be found as early as in 1965 [DP65]. Let us rephrase the (among others) theorem given by Derzko and Pfeffer:

Theorem 6 (The 4th Derzko-Pfeffer theorem).
Let $M, S \in \mathbb{C}^{n,n}$ where S is invertible. Then the following is always true:

$$\rho(M) \leq (1 - 1/n)^{1/2}\{(\epsilon(SMS^{-1}))^2 - |tr(M)|^2/n\} + |tr(M)|/n$$

where $\epsilon(A) = \sqrt{\sum_{i,j=1}^{n} |M_{i,j}|^2}$ is the Froebenius norm.

Now, using this theorem, let us attempt at constructing a noise matrix M. Setting $tr(M) = 0$ on the noise, and fixing S as the canonical basis we obtain:

$$\rho(M) \leq (1 - 1/n)^{1/2} \textstyle\sum_{i,j=1}^{n} |M_{i,j}|^2$$

Now, we can rely on **PSW** conjecture forcing $\rho(MD_{Id}^{-1}) < 1/2$ using a diagonal matrix D_{Id}:

$$2(1 - 1/n)^{1/2} \textstyle\sum_{i,j=1}^{n} |M_{i,j}|^2 \leq D$$

i.e, given a fixed dimension n and a fixed value D, we can properly bound the values of the noise matrix M such that the **PSW** Conjecture is respected. This can be done by carefully distributing the coefficients outside the diagonal.

However, the first thing to notice is that the bound is worse than the one given in **DRS** in most cases: the **DRS** bound is per vector, and this one is per matrix. Quick comparisons between the total sum of matrix coefficients will show the **DRS** bound is almost always superior.

Another theorem we could use on spectral radius is Gelfand's formula, which was also used in [PSW08]:

Theorem 7 (Gelfand's spectral radius formula).
$\rho(M) = \lim_{k\to\infty} \|M^k\|^{1/k}$

An extreme case would then to fix the limit to 0. This then warrants the uses of nilpotent matrices, i.e, matrices $M \in \mathbb{Z}^{n\times n}$ such that $\exists k > 0, M^k = 0$. We then need to have some form of generation for nilpotent matrices. One easy group of nilpotent matrices is the following:

$$\text{For } M = \begin{bmatrix} M_1 & \dots & M_1 \\ M_2 & \dots & M_2 \\ \vdots & \dots & \vdots \\ M_{n-1} & \dots & M_{n-1} \\ -\sum_{i=1}^{n-1} M_i & \dots & -\sum_{i=1}^{n-1} M_i \end{bmatrix}, M^2 = 0.$$

As the values M_i can be as large as wanted in this particular family, the **DRS** bound can be rapidly overblown, especially by the last row. We could also use other families of nilpotent matrices and combine them: The sum of nilpotent matrices being nilpotent, the space of possible noise could be large enough to ensure the security of cryptographic applications. However, it is unclear if using such matrices will allow efficient reductions: Large coefficients might hinder the convergence, and reaching a valid signature (if possible) might take unacceptable times for real-life cryptography. Furthermore, let us stress that the **PSW**-vector reduction is an approximation of Babai's rounding off algorithm: Thus, if the initial basis is "bad", then so could be the set of possible reduction results, i.e., having a noise with a zero-valued spectral radius is not enough. Therefore, a basis that is not diagonal dominant and have poor geometrical properties might not be suitable either.

Other approaches would be to remember that the spectral radius is the biggest eigenvalue (see Definition 44). Then we can attempt to use simple properties of the eigenvalues and control them to fix the exact value of the spectral value rather than bounding them. Let us look at the following: If M is a noise matrix, then $\rho(M)$ is the biggest value (in norm) that cancel the polynomial in $P(X) = \det(M - X_{Id})$. The literature on eigenvalues and their computations is extremely large [Wil65]: Bartel–Stewart [BS72], Hessenberg-Schur [GNVL79], Householder [Hou64], etc. It might be possible to reverse those methods and their subsequent works to construct a class of noise matrices respecting the **PSW**-conjecture. We also leave those studies for further work, as it likely requires much more studies. Overall, merging those approaches and the **DRS** approach into a uniform set of usable keys seems to be a widely open research question, let alone the computational practicability of those lattice classes (or subclasses).

For now, however, we provide practical efficiency tests on our new patch in the next subsection.

4.4.5 Setup Performance

Compared to the initial NIST submission where the code was seemingly made for clarity and not so much for performance, we wrote a modified version of **DRS** using NIST specifications and managed to have much higher performance. However, most of the performance upgrade from the initial code have nothing much to do with the algorithms of the **DRS** scheme: we did notice that most of the time taken by the **DRS** initial code was used for the conversion from the character arrays to integer matrices and vice-versa, which they had to do to respect the NIST specifications: the algebraic computations themselves were actually reasonably fast, considering the size of the objects manipulated.

This is the reason why we decided to isolate the secret matrix generation code from the rest of the initial original **DRS** code, in order to have a fair comparison between our own secret key generation algorithm to theirs. In that regard we choose to compare similar matrix sizes instead of similar security, as initial security estimates for the **DRS** submission were severely undermined by Yu and Ducas's recent discoveries and thus would lead to comparing efficiency on matrices with massively different sizes. Therefore we are making tests on the initial parameters of the **DRS** scheme. Looking purely at the secret key generation, we are indeed much slower, as shown in Table 4.1.

Table 4.1: Secret key generation time in milliseconds (average for 10^4 keys).

Dimension	912	1160	1518
*Old***DRS**	2.871	4.415	7.957
*New***DRS**	31.745	63.189	99.392

Note that we use the options $-march = native$ and $-Ofast$ which led us to use $AVX512$ instructions and other gcc optimization tweaks. The new setup is barely parallelizable as there is almost no code that can be vectorized which also explains the huge difference. While we wish to make a comparative performance to all other similar approaches, it seems the initial approach of **PSW** did not trigger further research and it remains an open topic, leaving **DRS** the only known fork of **PSW** to the best of our knowledge. Furthermore, timings were not provided in the original paper [PSW08]: A figure illustrating the evolution of the number of reduction loops was deemed sufficient to demonstrate its efficiency back in 2008.

Moreover, note that in theory, sampling randomly using our method should not be a problem while growing the size of our keys if we only consider the time complexity. The problem, however, concerns the amount of data to store (and the related memory accesses). The size of V_n grow more than exponentially and thus storing all related exact sizes could pose a problem for larger dimensions. A solution to drastically reduce the memory requirements would be to crop the extremely unlikely cases and round the remaining results, but so far this does not seem to be necessary for our largest parameters.

4.5 Security estimates

The goal of this section is to evaluate the security of the scheme. First computationally by measuring the effectiveness of heuristic key-recovery attacks, and then by discussing the potential structural weakness of choosing diagonally dominant matrices as our key structure.

4.5.1 BDD-Based Attack

Currently, the most efficient way to perform this attack will be, first, to transform a **BDD** problem into a $\mathbf{uSVP}_\gamma$ (Kannan's Embedding Technique [Kan87], assuming $v = (0, \ldots, 0, d, 0, \ldots, 0)$, and use lattice reduction techniques on the lattice spanned by $[v|1]$ and the rows of $[B|0]$. By using this method, we obtain a $uSVP$ with a gap

$$\begin{pmatrix} v & 1 \\ B & 0 \end{pmatrix}$$

and second to solve this new $\mathbf{uSVP}_\gamma$ using lattice reduction algorithm. By using this method, we obtain a $\mathbf{uSVP}_\gamma$ with a gap

$$\gamma \approx \frac{\Gamma\left(\frac{n+3}{2}\right)^{\frac{1}{n+1}} Det(\mathcal{L})^{\frac{1}{n+1}}}{\sqrt{\pi}\|M_1\|_2} \approx \frac{\Gamma\left(\frac{n+3}{2}\right)^{\frac{1}{n+1}} d^{n\frac{1}{n+1}}}{\sqrt{\pi}\|M_1\|_2}. \tag{4.1}$$

Lattice reduction methods are well studied and their strength are evaluated using the Hermite factor. Let $\mathcal{L}$ a $d-$dimensional lattice, the Hermite factor of a basis B of $\mathcal{L}$ is given by $\|B[1]\|_2/det(\mathcal{L})^{\frac{1}{n}}$. Consequently, lattice reduction algorithms strengths are given by the Hermite factor of their expected output basis. In [GN08], it was estimated that lattice reduction methods solve $\mathbf{uSVP}_\gamma$ with γ a fraction of the Hermite factor. We will use a conservative bound of $\frac{1}{4}$ for the ratio of the $\mathbf{uSVP}_\gamma$ gap to the Hermite factor. As we do not have a fixed euclidean norm for our secret vectors we have to rely on the approximates given to us by our new random method in sampling noise vectors M_i. In our case, we know that for any vector $v \in \mathbb{Z}^n$ we have $\|v\|_2 \geq \frac{\|v\|_1}{\sqrt{n}}$, and our experiments (as seen below) allow us to use a higher bound

$$\|v\|_2 \gtrapprox \sqrt{2}\frac{\|v\|_1}{\sqrt{n}}.$$

4.5.2 Expected Heuristic Security Strength

Different papers are giving some relations between the Hermite factor and the security parameter λ [vdPS13, HPS+17] often using BKZ simulation [CN11]. Aiming to be conservative, we are to assume a security of $2^{128}, 2^{192}, 2^{256}$ for a Hermite factor of $1.006^d, 1.005^d, 1.004^d$, respectively. We set $D = n$, pick hashed messages $h(m)$ such that $\log_2(\|h(m)\|_\infty) = 28$, $R = 24$ and $\Delta = 1$.

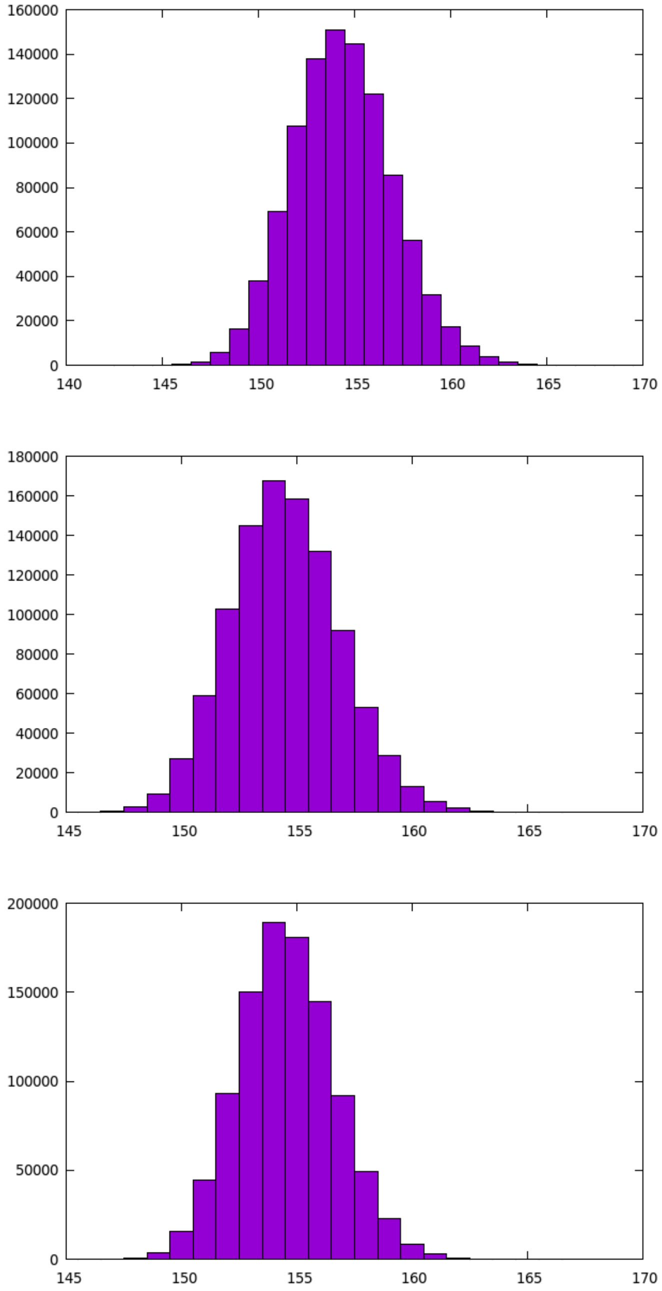
Table 4.2: Parameter Sets.

Dimension	Δ	R	δ	γ	2^λ
1108	1	24	28	$< \frac{1}{4}(1.006)^{d+1}$	2^{128}
1372	1	24	28	$< \frac{1}{4}(1.005)^{d+1}$	2^{192}
1779	1	24	28	$< \frac{1}{4}(1.004)^{d+1}$	2^{256}

Table 4.2 parameters have been choosen to obtain a $\mathbf{uSVP}_\gamma$ gap (Equation 4.1) with $\gamma < \frac{\delta^{d+1}}{4}$ for $\delta = 1.006, 1.005, 1.004$. Our experiments show us that the distribution of zeroes among sampled noise vectors form a Gaussian and so does the euclidean norm of noise vectors when picking our random elements x, x_i uniformly. Here we include below the distribution of 10^6 randomly generated noise vectors v with the x-axis representing $f(v) = \lfloor 100\sqrt{\frac{\|v\|_2^2}{D}} \rfloor$ where D is the signature bound (see Fig 4.5).

We can see that the generated noise vectors follow a Gaussian distribution as far as their norms are concerned, and we believe it makes guessing values much harder for an attacker should they choose to focus on finding specific values or vectors (as it was the case in the original attack from Yu and Ducas [YD18a]). We also conducted experiments, using BKZ20 from the fplll library [FPL] (see Fig 4.6). Without any surprise we notice our new setup is seemingly resistant around dimension 400, where conservative bounds led us to believe the break happen until approximately dimension 445. However the sample size is relatively small (yet computationally expensive to obtain) and thus should not be taken as a proof value, but rather as a heuristic support against heuristic attacks.

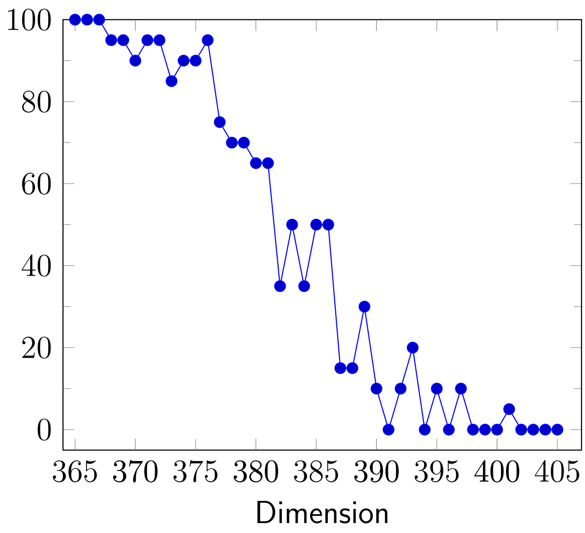
Figure 4.5: $f(v)$ distribution for $n = 1108, 1372, 1779$ and $D = n - 1$ over 10^6 samples



4.5.3 A note on the structure of diagonally dominant lattices

Since there has been a lot of discussion about provably secure schemes, especially for lattice-based schemes, one of the raised "flaws" of the **DRS** scheme compared to most of the other lattice-based submissions to the NIST PQC competition was the lack of a security proof. This subsection does not provide a security proof for

Figure 4.6: Percentage of key recoveries of BKZ20 (20 sample keys/dim)



this particular instantiation of the **DRS** scheme, however it aims to increase the confidence on the structure of diagonal dominant lattices. To do so, we are going to define a "new" problem, based on a previously well-known problem.

Definition 46 (D-Pertubated Shortest Basis Problem (**PSBP**$_D$))**.**
Let $P \in \mathbb{Z}^{n\times n}$ *such that* $P = \{P_1, ..., P_n\}$ *is known and a solution of* ***SBP*** *on* $\mathcal{L}(P)$*.*
Given a known bound D *and an unknown matrix* $M = \{M_1, ..., M_n\} \in \mathbb{Z}^{n\times n}$ *s.t:*

- $\forall i \in [1, n]$, $\|P_i\| > D \geq \|M_i\|$
- $\forall i \in [1, n]$, $\|P_i + M_i\| > \|P_i\| + \|M_i\|$

We set $P_M = \{P_1 + M_1, ..., P_n + M_n\}$*.*
Solve ***SBP*** *on* $\mathcal{L}(P_M)$ *(from a* ***HNF*** *or a different basis).*

The idea here is to determine whether it is hard or easy to recover some randomly added noise on a basis that we know is the shortest basis, and even to recompute a new shortest basis from a "close" one. It is clear that this problem is "easier" to the original **SBP** problem. But how much easier is still a mystery: it is possible they are actually equivalent but as far as we know we have not seen any evidence to prove it. This problem is known in academic folklore, although in an informal way and probably with slightly different statements. The inequalities we state specialize in our "special" **SBP** problem allow us to exclude several problematic cases:

- $\|P_i\| > D$ allow us to exclude $P = 0$ where the problem is just equivalent to **SBP**.
- $D \geq \|M_i\|$ prevents insanely large M where D does not matter.

- $\|P_i + M_i\| > \|P_i\| + \|M_i\|$ prevents heuristically easier cases.

The actual question here would be to determine whether the problem is easy or hard for specific structures and distributions of P and values of D, which we currently do not know and probably require a much deeper work. Heuristically, since P is known and M is bounded, the best way to recover M is to use Kannan's extension technique and solve $\mathbf{uSVP}_\gamma$ where some coefficients are known, i.e the coefficients of P[a]. To the best of our knowledge, there is however no guarantee that recovering M would actually solve $\mathbf{PSBP}_D$: it just recovers a basis "close" to the solution as M should be "shorter" than the known part P.

Note that if $D = \lambda_1/2$[b], then recovering M can be heuristically reduced to solving n successive instances of **BDD**, namely one per vector of P. We also stress that we did not define a particular norm here. To the best of our knowdlege, there is no work in the literature to determine if solving n instances of **BDD** with non-trivial relations between those instances is actually as hard as than solving one instance of **BDD** with no particular structure. It is unclear how many instantiations of lattice-based cryptosystems are concerned by this problem. Historically, it seems that whenever a structural weakness have been found, it was mostly due to the structure of the variable part M rather than the fixed part P. The first structural attack on **GGH** seems to reflect that [Ngu99] and so does the recent attack on **DRS** [YD18a]: it does not seem there was historically much concern on the public part P. This could be either credited on the luck (or foresight) of cryptographers, or maybe there is an underlying relation we have yet to see.

However, we stress again that there might be a significant difference between a randomly sampled basis (under any distribution) and a basis constructed from known coefficients and bounded noise. As far as **DRS** is concerned, recovering M and solving $\mathbf{PSBP}_D$ is considered to be the same problem. The way instantiations of **DRS** are created, recovering the secret key in **DRS** is actually solving very special instantiations of $\mathbf{PSBP}_D$.

Property 24 (Hardness of **DRS** key recovery)**.**
*Recovering the secret key $S = D_{Id} + M$ of a **DRS** lattice is heuristically the same as solving $\boldsymbol{PSBP_D}$ with $P = D_{Id}$ on $\mathcal{L}(S)$ for the norm l_1.*

Proof. Substitute P by D_{Id}. As all vectors of D_{Id} are trivially orthogonal it follows that D_{Id} is the **SBP** solution to $\mathcal{L}(D_{Id})$. All vectors of M are lower than D in l_1-

[a] Note: this is exactly how the heuristic security of **DRS** was evaluated.

[b] We can assume $\lambda_1 = P_1$, however this is not always true: see example 2.10

norm by construction (a **DRS** key is diagonal dominant). Heuristically, the secret key is the shortest basis of the public key lattice, thus giving us the result. □

Let us stress again that **DRS** is not equivalent to the general $\mathbf{PSBP}_D$: **DRS** instantiations are specific and could potentially be broken in polynomial time, but even then it would not affect the hardness of $\mathbf{PSBP}_D$. The original **GGH** also uses special instances of $\mathbf{PSBP}_D$: it uses keys $S = (\sqrt{n})_{Id} + M$ where $M \in [-4, 4]^{n \times n}$, i.e $\mathbf{PSBP}_D$ with norm l_∞, $D = 4$ and $P = (\sqrt{n})_{Id}$, and was yet to be "asymptotically" broken as far as key-recovery attacks were concerned. To the best of our knowledge, those attacks still run in exponential time. We would like to stress that this section actually did not cover the message security as [PSW08] actually points out that a full-key recovery is not necessary to forge signatures: as long as an attacker can find a **PSW**-good basis then the **PSW** vector reduction algorithm could converge.

This trivial analysis however showed that the security of key recovery attacks is, as expected, based on the noise and hopefully removes the concern about having large diagonal coefficients in a basis. If further work show that a noise matrix M is provably hard to recover under certain assumptions for $\mathbf{PSBP}_D$, then constructing **DRS** under those assumptions could make it provably secure (although this only concerns "exact" key-recovery attacks).

Note that, just like the attacks on **GGH**, the key recovery attack of Yu and Ducas is enabled by the recovery a large amount of tuples "messages-signatures" from the same key. The problem we just defined does not thwart the attack, as this is a problem related on a possible statistical leak given by specific noise coefficients, all the while using a particular signing algorithm. In short, an interesting research to thwart statistical heuristic attacks would be to find some form of pre-selection or/and noise structure where statistical independency can be proven to hold for each signature: from our understanding, this is actually a direction Yu and Ducas suggested to pursue [YD18a]. It is possible that the leak found by Yu and Ducas can be patched by modifying the signing algorithm without modifying the noise as we did. As of November 2019, an extended version of [YD18a] available in [YD18b] reduces the security of our original contribution [SPS19b]. While the updated attack is clearly not as strong as the previous attack, it still provides further motivation to deepen the research.

4.5.4 A small density comparison with ideal lattices

Using an ideal lattice as a noise reduce the available set of secret keys drastically. The main point of ideal lattices is to reduce the size of the public key and the computation costs. In that regard, given a principal ideal lattice $\mathcal{L}(g, f)$, f should be public for efficient computations to be available. $g^i \mod f$ by iterating over i should give the rest of the noise beyond the first vector. Assuming f is anti-cyclic, or chosen in a way where $g^i \mod f$ has a taxicab norm lower than some constant D for all i, then the noise structure is decided by the choice of the first vector g.

Basically, compared to an ideal lattice, we pick $n-1$ more vectors randomly. Which means that while the choice for p is at most $\|V_n\|$ possibilities, our noise set has a factor $\|V_n\|^{n-1}$ over ideal lattices. V_n being a set that grows more than exponentially as n increase, we can safely assume our noise set has a higher density than ideal lattices. It is unclear however if the density is exponentially vanishing for a fixed determinant as it is the case of ideal lattices: we do not know how to fix the determinant of a newly sampled **DRS** lattice unlike the previous chapter.

Nevertheless, we believe it is safe to claim that the structure used here is safer than the structure provided by ideal lattices which are currently quite popular. Our reasoning should also apply for module lattices in a lesser extent (but with a similar asymptotic scale).

Acknowledgements

We would like to thank Yang Yu, Léo Ducas and the anonymous reviewers for useful comments and suggestions.

Summary and conclusion

we presented in this chapter the **DRS** scheme, and another method to generate secret keys for providing experimental results on the statistical distribution of the keys generated following Yu and Ducas' attack. We demonstrate that our new approach is sufficient to improve **DRS** to be secure against machine learning attacks as reported earlier in the literature.

Chapter 5

Freivalds-accelerated Signature Verification

This chapter is an extended version of the short paper we published in ISPEC 2019. It is merely a consequence of **DRS** (Diagonal Reduction Signature scheme) not using a **HNF** (Hermite Normal Form), and could not be used for the NIST submission because of their required procedure for signatures. Let us remind some downsides about **DRS**:

- We do not have a **HNF** that can be computed in a fast manner as in less than a second, even MAGMA can compute it in at best 15 seconds for the largest parameters and we have yet to see an open-source code that performs faster on our parameters.
- The lack of **HNF** enforces us to compute some vector k to prove the membership of v in $\mathcal{L}(B)$, as the is some k s.t $kB = v$ if and only if $v \in \mathcal{L}(B)$.
- k is easy to compute with a **HNF** B, and so is kB. If B is not a **HNF** then even kB can be computationally expensive.

Dealing with the third point is the point of this chapter and the work we submitted at ISPEC: we aim to accelerate the verification process. To do so, we rely on the heuristic algorithm of Freivalds [Fre79]. Some reviewers commented that a key should never be reused. There is some merit in that belief, and if the reader agrees with that statement then the following chapter has very minor interest. Nevertheless the paper has been accepted after some peer-reviewed process and thus should still have some interest for some applications, especially if further work demonstrates a larger gain of efficiency.

We would like to thank the reviewers of both ARITH and ISPEC. Initially we submitted a draft to ARITH, which was maybe not clearly written and we would like to thank the reviewers for pointing out the lack of clarity. In this chapter, we attempt to clarify multiple points and hopefully make the work more understandable.

5.1 Freivalds' algorithm

Freivalds' algorithm (algorithm 22) for verifying matrix products [Fre79] is one of the first probabilistic algorithms to be introduced to show the efficiency and practicality of non-deterministic programs to solve decision problems over deterministic ones. Freivalds' technique had a major impact on several research fields and is still an active research topic to this date [DZ17, Dum18]. The decision problem solved by Freivalds is the following: given $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, three $n \times n$ matrices over an arbitrary ring $\mathcal{R}$,

can we verify that $A \times B = C$ with a faster method than recomputing $A \times B$?

Freivalds brought a probabilistic solution, which rely on a simple statement:

to check $A \times B = C$, we check instead $A \times (B \times v) = C \times v$

where v is a randomly sampled vector, and then it follows that the more we run this test, the more we decrease the probability of obtaining a false-positive.

This leads to Freivalds' algorithm 22 which is *perfectly complete*: it will always output ***TRUE*** whenever $A \times B = C$ is correct. It is also *sound*: the probability of outputting ***TRUE*** for $A \times B \neq C$ is negligible (as much as we want). An example MAGMA code can be found in the appendix (code of figure A.12). The gain in efficiency compared to a deterministic method is quite impactful as this shifts the arithmetical computations from a matrix-matrix product to a matrix-vector multiplication. We are not recalling the proof on the error probability bound for two reasons: the first reason is noticing it is actually an upper bound (independent of the matrices dimension and the entries of $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$) and already documented in [Fre79], and the second reason being the fact that we will later give a much tighter upper bound which will be more adapted to our case.

Example 15. *Testing $A \times B = C$ with Freivalds' algorithm.*

$$A = \begin{bmatrix} -2 & -2 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 2 & 0 & 1 \\ 2 & -1 & 2 & -1 & -2 \\ -2 & 2 & -2 & -1 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 1 & -2 & 1 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 & -2 \\ 1 & -1 & -2 & -1 & 2 \end{bmatrix}$$

Algorithm 22 Freivalds' algorithm

```
Require: $A, B, C \in \mathcal{R}^{n\times n}$, $f \in \mathbb{N}$ a failure probability
Ensure: Check the validity of $A \times B = C$ with a chance of false-positive under $2^{-f}$
i ← 0
while $i < f$ do
    $v \leftarrow$ Randomly taken in $\{-1, 1\}^n$
    $x \leftarrow Cv$, $y \leftarrow Bv$, $y \leftarrow Ay$
    if $x \neq y$ then return FALSE                ▷ check validity
    $i \leftarrow i + 1$
return TRUE
```

Let C, D two candidates for the product

$$C = \begin{bmatrix} -7 & -6 & 4 & 9 & 8 \\ 2 & 1 & 9 & 0 & 6 \\ 4 & -3 & -7 & 8 & -9 \\ 8 & -6 & 2 & -4 & 6 \\ 7 & -1 & 7 & -8 & 7 \end{bmatrix}, D = \begin{bmatrix} 2 & -3 & -6 & 6 & 4 \\ 1 & 0 & -2 & -2 & 1 \\ 1 & 0 & -2 & 4 & 3 \\ -4 & 1 & 6 & 3 & 1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix}$$

Sample a random vector, here for example $v = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 \end{bmatrix}$ and compute

$$v_{AB} \leftarrow v \times A = \begin{bmatrix} 6 & -1 & 5 & 0 & -2 \end{bmatrix} \textit{ then}$$
$$v_{AB} \leftarrow v_{AB} \times B = \begin{bmatrix} -2 & 6 & 10 & 1 & 3 \end{bmatrix}$$

Test now $v \times C$ and $v \times D$ with v_{AB}:

$$v_C \leftarrow v \times C = \begin{bmatrix} 14 & -1 & -7 & 3 & -12 \end{bmatrix}$$
$$v_D \leftarrow v \times D = \begin{bmatrix} -2 & 6 & 10 & 1 & 3 \end{bmatrix}$$

Conclusion C is wrong, D might be correct but it is not yet guaranteed: for example

$$M = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -3 \end{bmatrix}$$

Also gives $v \times M = v_{AB}$ however it is clear $M \neq A \times B$.

5.2 Modifiying Freivalds' technique for lattice-based signature verification

5.2.1 The first core idea: Modification of Freivalds' algorithm

In this work, we modify Frivalds' technique to obtain a faster probabilistic verification algorithm. The vector pairs (k, m) to check are not given by the person who needs the verification but by the signatory, and so is P_{key}. In the case where one public key is re-used over multiple message-signature exchanges, the equality $hP_{key} = m - s = v$ must stand true for all vector triplets (h, s, m) provided. In that case, we can introduce a random vector $x^\top$ and $X = P_{key} \times x^\top$ such that

$$(hP_{key} = m - s) \implies (h(P_{key} \times x^\top) = v \times x^\top) \implies (h \times X = v \times x^\top)$$

which reduce a matrix-vector multiplication check to the comparison of two scalar products (vector-vector multiplications).

A difference with the original Frivalds' algorithm is that we don't use $x \in \{-1, 1\}^n$ but rather, we will choose a prime p such that $x \in \mathbb{F}_p^n$ is taken randomly, and project the whole equation over the field $\mathbb{F}_p$. For sufficiently many vectors x, X and primes p, let's say k primes and k vectors x, our new validity condition is then

$$\begin{gathered} h \times X_1 = v \times x_1^\top \mod p_1 \\ \vdots \\ h \times X_k = v \times x_k^\top \mod p_k \end{gathered}$$

Note that projecting Freivalds' algorithm over a finite field was proposed in [KS93] for a non-cryptographic purpose, however to the best of our knowledge there is no work that modify the algorithm in the manner we just described.

We justify the choice to use multiple different moduli as we are in a cryptographic application and thus want to ward off potential attacks, this will be expanded in section 5.3.

Let us compute the probability of failure of our verification algorithm. First of all, the algorithm is *perfectly complete*, i.e it will never output a false negative. The last thing to check is then the probability of a false positive. In that regard, rather than thinking of probability of a false positive, let us first compute the proportion of positive results over all possibilities given a prime p. Let us enumerate all possibilities. If v is fixed, then $v * x^\top = a_p \mod p$ is also fixed. So the proportion of couples (h, m) giving a positive result is the probability of

$$h \times X = a_p \mod p$$
$$\sum_{i=1}^{n} h[i]X[i] = a_p \mod p$$

Without losing generality, if we choose and index j such that $X[j]$ is non-zero (X being obviously chosen non-zero) and fix every other coefficient h_i such that $b_p = a_p(\sum_{i \neq j} h[i]X[i])^{-1} \mod p$, we obtain the result of a positive output with the same proportion as verifying

$$h[i] = c_p \mod p$$

which is $1/p$ for a given prime p. As this reasoning is sound for any $v = m - s$ and in any triplet (h, m, s), we determine the quantity of false positives being the difference between the amount of positive outcomes and the amount of valid positive outcomes, which set a proportion of false positives of being strictly under $1/p$ (and by extension its probability over all possible samples).

If we repeat this process over k different vectors, the false positive probability lowers to below p^{-k}. Generally speaking, if we try the test once per couple prime/vectors over k primes $\{p_i, x_i\}_{i \in [1,k]}$, then the probability of obtaining a false positive becomes lower than $\prod_{i=1}^{k} p_i^{-1}$. This is a tighter upper bound over Freivalds' initial upper bound, although our work only concerns our very specific case and does not apply to the general scope of Freivalds' technique.

Note that quite naturally, there are good reasons this modification have been avoided in other applications: this modification makes the whole process overall slower when verifying one signature. After all, we introduce a new scalar product on the right-hand side $m - s$ and a supplementary scalar product on the left-hand side hP_{key}, with overall no gain in complexity aside from an equality test among integers rather than vectors. We can safely assume that in general a multiplication of scalars is more expensive than an equality test of vectors, thus this change is only beneficial when $P_{key} \times x^{\top}$ is computed once, which is the point of our new verifier presented in the next subsection.

5.2.2 The second core idea: Changing the verifier

With our previous idea in mind, we need to explain what we aim to modify in the previous **DRS** scheme. First let us briefly recall how the sender/verifier Alice and the signatory Bob acts in 5 steps:

1. Bob generates a pair of keys $\{P_{key}, S_{key}\}$.

2. Bob keeps the secret key S_{key}, and sends the public key P_{key} to Alice.
3. Alice sends a random vector m with "large" norm $\|m\|_\infty$ to Bob.
4. Bob uses S_{key} to send the signature $\{h, s\}$ to Alice.
5. Alice verifies that $\|s\|_\infty$ is "low" and $hP_{key} = v = m - s$.

While we can consider the verification process to be entrusted to a third-party like a certification authority, here we restrict ourselves on exclusively modifying the computation of the verification which is step 5, and inserting a precomputation which can be placed after or during step 2.

One important point to stress on is that Alice does not need to communicate to Bob she is using a precomputation. The whole process is oblivious to Bob and his role does not change at all compared with the existing **DRS** process. Hence, it seems natural for us to assume Alice will keep her computations secret as there is no apparent benefit in revealing them.

Precomputation

The precomputation construct the samples required to apply our modified Freivalds' test and can be described in two halves as follows:

- Generate a family of tuples $(p_i, x_i)_{i\in[1,k]}$
- Compute $T = (p_i, x_i, X_i)_{i\in[1,k]}$ where $X_i = P_{key}x_i \mod p_i$ given P_{key}

The first half of the precomputation do not requires input from Bob as the dimension is supposed to be public, therefore those can even be precomputed before Bob generating his keys in step 1. The choice of random generators for primes and vectors are important for security and efficiency considerations, however those are not the main point of the paper. As far as our experimental results are concerned, we just used the basic random function "**rand()**" of the library "**stdlib.h**" in C with the classical modulo operator % to generate our vectors, and our primes are randomly taken in a set we will discuss in the security section of this paper, using the MAGMA software [BCP97] to pick primes and write them into a header file used by our code before the compilation. Its computation time is negligeable compared to the second part of the precomputation which involves matrix-vectors modular multiplications.

In the second half of the precomputation, Alice does not need to store P_{key} at the end, and furthermore she does not even need to store the whole public key *while computing* X_i. Since for each row j of P_{key} Alice can independently compute $P_{key}[j] * x_i = X_i[j] \mod p_i$, Alice can discard every row of P_{key} where the corresponding computation is finished and choose to only receive a certain amount of rows at a time, which would reduce the amount of internal memory required for the whole precomputation (and allow for further parallelism). The cost of the second half of the precomputation is the main cost of the whole precomputation process.

Since we are talking about the possibility of delayed communication (i.e chunks of data are not sent in one single time, we can think of a real-life slow WiFi), then another possibility here arises as further precomputations. This is a precomputation which can be done before or after the hashed vector to reduce/sign is provided to Bob. Since we have to compute $k \times B \times x^\top = (m - s) \times x^\top = (m \times x^\top) - (s \times x^\top)$, we can compute $m \times x^\top$ before the signature of m i.e (k, s) is returned. While this seems like we are multiplying the cost by 2 on the right hand side, note that in the case of delayed communications we can afford this extra computation. Note that s has low size, therefore upon receiving s from Bob $s \times x^\top$ should be much faster to compute than $(m - s) \times x^\top$ whenever machine word sizes are low.

This latter technique is not used in our experiments nor was mentioned in our original paper, but we believe it is another nice application. For now we will present the basic idea of our new verification method.

New verification method

The new verification method will apply our modification on Freivalds' algorithm using our precomputation step. Alice, at step 5, previously discarded the public key P_{key} and kept some small footprint in the form of a secret list T of triplets and sent a random message m. As she received in step 4 the signature (k, s) from Bob, her verification process is now described by algorithm 23.

This new verification algorithm is more compact than the original one and also simpler to understand. The only remaining point to deal with is to choose how large h and the primes p_i need to be. We will discuss that in the next section when discussing security.

Example 16. *Let us use the example of the previous chapter. Bob has a public key* P.

Algorithm 23 New Verification

```
Require:
    a list of triplets $T = (p_i, x_i, X_i)_{i \in [1,k]}$ and a message $m$ from Alice
    a signature $(h, s)$ from Bob
    a public bound $D$ on the signature norm
Ensure:
    a boolean $R$ stating whether $(h, s)$ is a valid signature for $m$ and $P_{key}$
    $R$ is a false-positive with probability strictly less than $\prod_{i=1}^{k} p^{-1}$

$R \leftarrow \{\|s\|_\infty < D\}$                ▷ Verifies the max norm of the signature
for $i \in [1, k]$ do
    $R \leftarrow R \wedge \{hx_i = (m - s)x_i \mod p_i\}$      ▷ Verifies modular equalities
return $R$
```

$$P = \begin{bmatrix} -1840 & 2471 & -382 & -820 & 710 & 3048 \\ 1966 & -1378 & 1486 & 1721 & 1430 & -4090 \\ -1998 & 4317 & 994 & 271 & 3660 & 2211 \\ 2729 & -3460 & 746 & 1375 & -680 & -4662 \\ 2784 & -6566 & -1866 & -801 & -6100 & -2700 \\ 3679 & -3323 & 2144 & 2716 & 1380 & -7160 \end{bmatrix}$$

Alice does not store the whole public key, and instead sample 2 *primes* $a = 2, b = 3$ *and random vectors* $x_a \in \mathbb{Z}_a^n$ *and* $x_b \in \mathbb{Z}_b^n$.

$$x_a = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \textit{ and } x_b = \begin{bmatrix} 2 & 2 & 1 & 0 & 2 & 2 \end{bmatrix}$$

We then compute $X_a = P \times x_a^\top \mod a$, $X_b = P \times x_b^\top \mod b$

$$X_a^\top = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \textit{ and } X_b^\top = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 2 \end{bmatrix}$$

Alice sends a message which hash to a vector v *to reduce and Bob sends a signature* (k, w).

$$k = \begin{bmatrix} -54029 & -77227 & 6908 & -38654 & -4594 & 50148 \end{bmatrix}$$

$$v = \begin{bmatrix} 924 & 232 & 131 & 692 & 439 & 694 \end{bmatrix}$$

$$w = \begin{bmatrix} 0 & 9 & -9 & -1 & -1 & 0 \end{bmatrix}$$

Now Alice wants to verify with a certain probability that the signature is correct. The first time is to check $\|w\| < 10$. *This is easily seen. The second is to test that* $kP = v - w$.

$$k \times X_a^\top \mod a = 6908 \mod 2 = 0 \textit{ and } k \times X_b^\top \mod b = -854 \mod 3 = 1$$

$$(v - w) \times x_a^\top \mod a = 1274 \mod 2 = 0 \textit{ and } (v - w) \times x_b^\top \mod b = 4702 \mod 3 = 1$$

Alice is now sure with an error probability lower than $1/(a \times b)$ *that Bob gave a correct signature.*

Note that the above example had lots of zeroes. This is due from us picking only two primes a, b of extremely small sizes for the sake of the example. By picking a large number of large primes this phenomena is essentially discarded. A full working code of the previous example can be found in the appendix A.13, where primes taken are a bit more random. As we will see later, it does not actually matter much that a and b are prime. The main reason the original paper was focused solely on using primes was the fact that is was the easiest option to explain in a small amount of words and we found ourselves quite limited in the amount of information we could fit. We just chose them prime for simplicity but as far as efficient results are concerned, or even security, we will suggest other alternatives.

For now and consistency with the paper we published, we will focus on the heaviest option, i.e considering a list of large prime numbers, and talk about more reasonable alternatives later.

5.3 Security considerations

While the previous attacks on the old **DRS** are well-understood heuristics relying either on machine learning [YD18a] or pure lattice reduction as with most other lattice-based schemes (being signature-based or decryption-based), *our modification does not thwart previous attacks nor does it reinforce them and thus rely on the same security assumptions.*

However, this is only when considering the only secret was the diagonal dominant matrix S_{key}. Here, we introduce a new secret, which is the list of triplets T generated by Alice. Thus, new attacks venues can be considered which, to the best of our knowledge, were also not considered in others lattice schemes submitted to the NIST. We will consider them in this section. We briefly present the two avenues we found and explain our reasoning on why only the second can be considered, and tackle this issue. Note that our reasoning discard all attack venues that can affect the old **DRS** independently of our new method, as this would be out of this paper' scope, and we stress it is hard to construct a security proof when an attack aside from exhaustive search cannot be constructed.

5.3.1 Attack models

A malicious Bob

One attack is to try to guess the triplets generated by Alice, as malicious Bob, by sending carefully crafted keys and signatures. While it is definitively an interesting idea, as long as Alice generates a different triplet for each public key (using a hash of P_{key} as a seed for example) and only answers $True$ or $False$ in the verification, we do not see any gain malicious Bob could have over a honest Bob.

A honest Bob, and a "fake Bob" Eve

To the best of our knowledge, the only other attack venue is having a honest Bob, who is giving good signatures, and Eve, who has no knowledge of the secret key S_{key}, wanting to sign as well as Bob but could not in the existing **DRS** scheme. Let us suppose Eve knows that Alice is using our technique for signature verification, although assuming she has no knowledge of the triplets T and knows as much as Alice concerning Bob. Can Eve make Alice believe Eve is Bob?

To this purpose, we assume Eve has to generate a false-positive from Alice's verification algorithm. As Alice can make the primes p_i and their quantity k as large as she wants, it seems unreasonable to assume Eve can randomly fall into the $\prod_{i=1}^{k} p_i^{-1}$ false-positive probability. Eve cannot resort either to a lattice-reduction technique on an easier lattice stemming from T if she has no knowledge of T. Furthermore, building a false-positive for the modified Freivalds' test is not enough: one has to guarantee the vector-signature s is short enough. It is then possible that the number of false-positives drastically decreases, which reinforces the security of our modification however counting the number of false-positives within a bound seems non-trivial.

Therefore, we believe that for Eve to be successful she must at least recover T fully. *It is unclear if the knowledge of the primes p_i is enough for Eve to recover the associated vectors x_i. While this is a very obvious overexaggeration on Eve's attack capabilities, we will assume for a simpler analysis that guessing exactly all p_i is sufficient to trigger a false-positive on Alice's side.* We will now discuss this choice and explain in the next section how to alleviate this (potential) issue of Eve recovering T.

Considerations behind our "hidden moduli" choice

Very simplistically, when one crafts an algorithm on a particular mathematical structure, they usually know the result of an operation "+" as $c = a + b$ given a, b. Crafting an algorithm when having little knowledge on the result of basic operations as $c = a + b$ is trivially more challenging and by hiding the moduli ring in which computations are done is a way to achieve a certain sense of security. Note that even if the moduli are known, the security is by no means compromised. Let us state the ideas of the overall problem:

Definition 47 (Hidden Modular Vector Product).
Let x be a unknown matrix $x^\top \in \mathcal{R} = \mathbb{Z}_{m_1}^{l_1} \times \ldots \times \mathbb{Z}_{m_n}^{l_n}$ and M a known matrix $M \in \mathbb{Z}^{N \times N}$ such that $N = \sum_{i=1}^{n} l_i$.
Let Φ be an oracle that given two vectors (k, v) outputs the result of the equality test $\Phi_{M,x}(k, v) \rightarrow (k \times M \times x^\top \text{ is equal to } v \times x^\top)$.
Before each query to the oracle, you are given a single vector v'. The query must then be of the form $\Phi_{M,x}(k, v'' - v')$ where v'' is small otherwise Φ defaults to ***FALSE****.*
The problems are the following:

- *How many calls to Φ are necessary to forge a* ***TRUE*** *output with $k \times M \neq v'' - v'$?*
- *Related problem: How many queries to Φ is necessary to recover x?*

Clearly, the difficulty depends of the number of numbers (l_i, m_i) and the matrix M, but the problem might not actually be hard. Note that if you can solve $\mathbf{GDD}_\gamma$ on $\mathcal{L}(M)$ for any given vector v', then creating valid instances $(k, v' - v'')$ is trivial. The problem can be hard in forging the first **TRUE** case. Our application, however, obtain multiple **TRUE** cases.

In our work we solely focus on the simpler problem: obtaining an invalid **TRUE** which does not solve the underlying $\mathbf{GDD}_\gamma$ problem on $\mathcal{L}(M)$ once is enough to break the scheme. This could happen if the keys x, M are not refreshed often "enough". But how much is "enough"? Note that if some queries $(k_1, v_1), \ldots, (k_n, v_n)$ reveal themselves to be true (independently of whether $k \times M \neq v'' - v'$ is verified), then you could potentially construct other solutions from $< (k_1, v_1), \ldots, (k_n, v_n) >_{\mathbb{Z}}$.

Thus, once we obtain a **TRUE** output a polynomial amount of times, the "related problem" reveals some structure by linear algebra by constructing invertible matrices K, V over $\mathcal{R}$ s.t

$$K \times M \times x^\top = V \times x^\top \text{ i.e } V^{-1} \times K \times M \times x^\top = x^\top$$

i.e gathering systems of equations of the form $A \times x^\top = x^\top$ where multiple possible matrices A known. It is still unclear if this is sufficient to either guess x or forge invalid signatures $k \times M \neq v'' - v'$ which gives $\Phi_{M,x}(k, v'' - v') = \mathbf{TRUE}$, but it is a structure to consider.

Our problem is a harder version of the previous problem: while $l_1, ..., l_n$ can be given as public, $m_1, ..., m_n$ are hidden. This lead to the following problem:

Definition 48 (Hidden Modular Vector Product with hidden modulis)**.**
Same as previously but the ring $\mathcal{R} = \mathbb{Z}_{m_1}^{l_1} \times ... \times \mathbb{Z}_{m_n}^{l_n}$ now have hidden moduli.

While in the first problem, if we could solve the first problem a sufficient amount of times, the "related problem" might have been solvable in some way, this case could be possibly much harder as a modular inverse is no longer easily available (nor are any modular operations), although this requires verification.

Nevertheless, one fact remains true: if some queries $(k_1, v_1), ..., (k_n, v_n)$ reveal themselves to output **TRUE**, then you could use $< (k_1, v_1), ..., (k_n, v_n) >_\mathbb{Z}$ to forge solutions. And thus our aim is to base the hardness of our modification on the original hardness of **DRS** or most lattice problems: our security assumption here is that the latter problem is not easier than solving $\mathbf{GDD}_\gamma$ over $\mathcal{L}(M)$ when the number of queries is "reasonably" bounded. How we "bound" that number of queries can be determined at the users' discretion: we do not provide a security proof, nor did we forge a heuristic attack to test its security. As long as a desired efficiency gain is achieved, then keys can be regenerated. We will show from experimental results on an inefficient assumption (picking random prime numbers as moduli, i.e m_i are prime and $l_i = 1$) that not even 400 signatures are needed for an overall speed efficiency gain on the largest parameters.

5.3.2 How to choose the primes

In order to dissuade Eve from trying to guess the correct set of primes $T_p = (p_i)_{i \in [1,k]}$, we have to make sure the number of possibilities is large enough. In that regard, *we are considering two objectives: one is to reduce the complexity of arithmetical operations used during the verification algorithm, and the other is to match a chosen level of security.*

Which naturally brings us to a natural question: is it easier to trigger a random false-positive, i.e to try our luck with an attacker's success of $\prod_{i=1}^{k} p_i^{-1}$, or to guess T_p? As we will observe later, the set of combinations T_p is picked from is actually far below $\prod_{i=1}^{k} p_i$. We also choose primes to obtain efficient arithmetic. To deal with the second objective, we will just fix the level of security to match the same level of security the original **DRS** algorithm was aiming to achieve with lattices of dimension $\{1108, 1372, 1779\}$: the NIST security levels $\{3, 4, 5\}$ which is basically requiring $\{128, 192, 256\}$ bits of security.

To reach that number, let us present how we determine the number of combinations available when choosing primes of a certain amount of bits. Suppose we have a set S of primes, and pick k primes from it which gives $\binom{S}{k}$ combinations to choose from. We now have to determine both S and k. To give an idea of the numbers required, we give table 5.1 and 5.2 and refer to a table available online [eS18] referencing the number of primes.

Table 5.1: Number of primes per bit size

#Bits	20	21	22	23	24	25	26	27	28	29	30	31	32
#Primes	38,635	73,586	140,336	268,216	513,708	985,818	1,894,120	3,645,744	7,027,290	13,561,907	26,207,278	50,697,537	98,182,656

Table 5.2: Size of set S necessary to achieve $\binom{S}{k} > 2^b$

k b	5	6	7	8	9	10	11	12	13
128	132,496,421	7,910,346	1,080,111	-	-	-			
192	-	-	610,573,333	63,155,327	10,957,838	2,727,426			
256							49,751,158	13,974,454	4,801,557

While taking low-bits primes to minimize the amount of modular reductions allows for more efficient arithmetic (see "lazy reductions" in Seiler's work for NewHope [Sei18]), we will very soon show that we have to combine multiple sets of large sizes of primes to achieve a reasonable amount of security. **DRS** fitted every computation within 64-bits for both speed and convenience, and ideally we should follow that philosophy.

There are four simple reasoning steps to keep operations within 64-bits:

1. $(k \times X)$ and $(m \times x)$ must hold in 63-bits (unsigned)

2. h, X, m, x have a dimension of 11-bits (DRS dimensions requirement)

3. $||k||_\infty \times ||X||_\infty$ must be within 52-bits

4. $||k||_\infty$ must hold in 26-bits, and so does p.

Note that Barrett's reduction [Bar86] is inapplicable here as it will overflow. Trying to reduce the number of modular reductions by reducing at the critical worst-case bound reached is a common technique used in some other cryptosystems and is sometimes called "lazy reduction" [Sei18]. Our operations can be represented as simply as a scalar product. Therefore, if our integers are in b bits, we need to apply a modular reduction every minimal amount l of multiply-and-add such that $63 = \log_2(l) + 2b$. Using signed representation, we can also gain one bit but it could increase the cost of the precomputation and we didn't test it.

In order to approximate the efficiency gained, we assimilate the cost of a modular reduction to one of a multiplication, which is roughly the same as a Barrett reduction. Keeping those in mind, we here present our choices of prime sets per NIST security level:

3 : 128-bits security: $k = 6$ with S the set of 28-bits primes

4 : 192-bits security: $k = 9$ with S the set of 27 and 28-bits primes

5 : 256-bits security: $k = 12$ with S the set of 24 to 28-bits primes

How we default to those choices is explained below:

Security level 2^{128} and lattice dimension 1108

We note that even using 31-bit primes and below will not give us enough integers in the pool X if we hope to use $k = 5$.

To achieve $k = 6$, we would need to use primes of at least 28-bits. In that regard, using 28-bits primes will force us to use modular reduction every 2^7 i.e 128 multiply-and-add operation or 256 using the trick described above, for a minimum of 4 modular reduction per scalar product i.e 8 per moduli for a total of 48 modular reductions.

On the other hand if we go for $k = 7$ we can use 25-bits and 24-bits primes (their combined sum is over the minimum value required) and verify that $24 * 7 > 128$ and no modular reduction is actually needed except for the checking phase. However one more moduli adds $1108 * 2 = 2216$ extra multiply-and-add operations and thus we consider it more efficient to have $k = 6$ with 28-bits primes.

Security level 2^{192} and lattice dimension 1372

We note that even using up to 32-bits primes in this case do not give us a big enough pool to use $k = 7$.

Using $k = 8$ forces us to use the pool of 31-bits and 30-bits primes. Using 31-bits primes force us to use a modular reduction at least every 4 multiply-and-add, which leads to 343 modular reduction per vector for a total of 686 per prime for a total of 5488 modular reductions.

Using $k = 9$, we have 90 modular reductions using the same reasoning on 28-bits integers (added to the pool of 27-bits integers) at the cost of 2744 extra multiply-and-add compared to $k = 8$.

Using $k = 10$ will lead to use 26 and 25-bits integers removing the need of modular reductions during the computation of the scalar product, however adding another extra 2744 operations does not seem to be an optimal choice.

Security level 2^{256} and lattice dimension 1779

Clearly $k = 10$ is not possible considering our previous security assumptions.

$k = 11$ is not possible without using 31-bits integers and lead us to 888 modular reduction per prime for a total of 9768 modular reductions.

$k = 12$ is possible when we use all primes ranging from 28-bits to 24-bits included: that gives 14 modular reduction per prime i.e 168 in total, for 3558 extra multiply-and-add.

5.4 Implementation results

5.4.1 Time results on a basic implementation

To make a fair comparison, we first give the time given by the original algorithm (setup, sign and verify). Time is given as an average in milliseconds and computations were done using a Intel(R) Xeon(R) Gold 6128 CPU @ 3.40GHz processor using a non-optimized C implementation and re-using the code provided by the our own original submission on the NIST website (see table 5.3).

We then showcase the base case where we do not take account of the number of

Table 5.3: Average time (in ms) for the existing **DRS** scheme

Phase Security	Setup	Signature	Verification
128	67.5	1.495	0.89
192	102	2.46	1.68
256	162.9	3.82	3.50

combinations and use just enough 32-bit integers to reach the product size needed, and compare them with our choices with smaller primes (28-bits or less) in a larger amount to reach the combination size needed (see table 5.4). Note that the generation of primes is not included, as we used MAGMA [BCP97] to pick primes and write them into a header file used by our code before the compilation. Picking primes, however did not take any significant amount of time (almost always lower than 10ms), and we used an external software (namely MAGMA) to select them. Following this, we do not think reporting the time taken for the prime generation is very relevant, as the literature also points out it is on a much lower scale than a matrix-vector multiplication (for our sizes, see [Mau95] and subsequent work on either heuristic or deterministic algorithms).

Table 5.4: Average time (in ms) for the precomputation/verification algorithms

Security Phase	128	192	256
Precompute (32-bits)	100.16	223.9	646.69
Verify (32-bits)	0.1328	0.2515	0.4297
Precompute (28-bits)	153.21	363.1	1005.1
Verify (28-bits)	0.1048	0.2006	0.3429

We observe that the precomputation is heavier than the generation of the keys. Which is expected as we are dealing with multiple modular matrix-vectors multiplications, whereas the original **DRS** setup only had to deal with randomized vectors additions. The number of signatures generated per key to break even in time (*including precomputation*) compared with the old **DRS** is reached for 256-bits of security with 319 signatures (28-bits case) and 211 (32-bits case), and even less signatures for the lower levels of security. Table 5.5 shows the number of signatures to break even for each parameter set.

5.4.2 Memory storage

As we mentioned previously, Alice does not need to store Pk in our alternative scheme. Memory-wise, this showcases an obvious advantage for the verifier to re-

Table 5.5: Average number of signature verifications to break even in overall time efficiency

Security	128	192	256
Signatures (28-bits)	196	246	319
Signatures (32-bits)	133	157	211

quire only a quasi-linear amount of memory in function of the dimension rather than a quadratic amount (i.e full public key). Here in all 3 cases we store k prime integers of 28-bits, plus $2 * k$ vectors of dimension n containing 28-bits integers, thus the memory taken in bytes is $\lceil 28(k + 2kn)/8 \rceil$ (see table 5.6).

Table 5.6: Memory storage in bytes for P_{key} and its footprint T

Security	128	192	256
P_k the public key	7,672,900	11,764,900	19,780,257
p_i, x_i, X_i	46,557	86,468	149,478

Another potential worry in term of memory in our modified scheme is that the prime number generation might be taking a lot of memory for the verifier. However, after decades of research on prime number generation we do not believe this is really the case [JP06]. Furthermore, we do not actually need primes for this approach to be usable: we just need the integers to be coprime, which brings a larger set of available integers at a much lower size.

5.5 Beyond DRS and the usage of primes

5.5.1 Any random integer ring is fine

The first point we want to stress is that we do not need to actually generate primes. We only need a large set of large factorizable integer rings. Basically speaking we want a set S of integer rings where the following hold:

$$\text{For } \lambda\text{-bits of security we need } |S| > 2^{\lambda}, \text{ and } \forall \mathcal{R} \in S, |\mathcal{R}| > 2^{\lambda}$$

However for both security and efficiency assumption, we need to assume on top of the previous condition:

$$\exists k \in \mathbb{N}, \text{ such that } \forall \mathcal{R} \in S,\ \mathcal{R} \equiv \mathcal{R}_1 \times \ldots \times \mathcal{R}_k \text{ where } \forall i \in [1, k],\ |\mathcal{R}_i| < 2^{32}.$$

The lower k and the $|\mathcal{R}_i|$, the more efficient it is. Such large groups exists by CRT decompositions, and asking for such large sets also requires large sets where computations are easy. The RNS easily come to mind as those, introduced by Garner

[Gar59], already enjoy multiple applications in cryptography [BI04, BP04, BEM16, BEHZ16].

RNS can work for computations over bounded integers of any size, as long as the final result can be recovered and is guaranteed to exist within the bound. Therefore, another alternative is the following:

- Use a public RNS basis commonly shared by all parties, or just by a single signatory.
- Compute operations within a ring $\mathcal{R}$ hidden such that $|\mathcal{R}|$ is coprime with every RNS component (known in advance).

While it might seem surprising for non-arithmeticians to use another ring $\mathcal{R}_{\text{RNS}}$ to do computations within a ring $\mathcal{R}_{\text{hidden}}$, there have been previous cases where RNS was used as a mean to accelerate computations within **RSA** [BI04]. It is clear using this technique drastically increases the set of available hidden rings at almost no cost, however the reconstruction or basis change operations, if needed, can be costly.

At this point no experimentations were done, but intuitively it seems clear that both approaches will lead to better results both in efficiency and security.

5.5.2 Other types of basis and lattice-based signature schemes

Remember that for any lattice, $v \in \mathcal{L}(B)$ if and only if there exists k such that $kB = v$. In the case of **DRS**, both k and v are part of the signature, but for most lattice signature schemes k is not provided. The reason is simple: most lattice-based schemes use a **HNF**, and k can be computed in polynomial time if B is a **HNF**, and even in linear time when B is a perfect **HNF**. Thus, k is computed “on the fly” during the verification process. We provide the two cases below

Perfect **HNF** basis $B \in \mathbb{Z}^{n \times n}$	Cryptographic q-ary basis $B \in \mathbb{Z}^{n \times n}$
$kB = v$ gives “$\forall i > 1$, $k_i = v_i$”.	$kB = v$ gives “$\forall i$, $B_{i,i} = 1 \implies k_i = v_i$”

In both cases, when $k_i \neq v_i$, a modular reduction is applied to compute k_i.

If our modification is to be applied to another lattice-based scheme, then k must be included as a part of the signature. This will increase the size of the signature significantly while also adding computational costs for the signatory. The gain is therefore less obvious: the verifier does not need to store the public key, but might need to deal with a vector k which is potentially very large. While verification costs can indeed be decreased, communication costs will increase as a result on top of the signature size and cost.

Furthermore, most recent lattice-based schemes do not have efficiency problems as they rely on a ring structure. The security of schemes based on ring-based lattices is maybe debatable but their efficiency is a strong selling point and might not need further efficiency gains from this technique. Even without using the ring structure, the case of q-ary lattices where q is low or a power of 2 already enjoy fast arithmetic that can be lost by applying our modification. In short, plugging our modification into another existing cryptosystem requires a case-by-case analysis and efficiency gain are not necessarily apparent.

However, there is also positive points to consider in our modification when applied to **HNF** basis:

- Part of k can be removed from communications as $k_i = v_i$ for many positions.
- Precomputation of $B \times x^\top$ when B is in **HNF** is faster than with random matrices.

The first point is straightforward to understand. The second point give some relations in $B \times x^\top$:

- In the case of perfect **HNF**, $X = B \times x^\top$ gives $X_i = B_{i,1} \times x_1 + x_i$.
- In the case of cryptographic q-ary lattices, there is some integer m such that $i \leq m \implies X_i = q \times x_i$ and $i > m \implies X_i = x_i + \prod_{j=1}^{m} B_{i,j} x_j$

We can observe that the case of perfect **HNF** leads to an extremely fast precomputation phase. x_1 can even be fixed to 0 to skip most of the precomputation. While not being as efficient as the perfect **HNF** case, the q-ary case also gives a lot of easy precomputations compared to the **DRS** case.

At this moment we did not conduct any experiments to test the efficiency of those cases. As mentioned before, this is a case-by-case basis which requires much further work.

Summary and conclusion

We introduced a modification of Freivalds' algorithm to introduce a faster verification method to **DRS**. This process is done while not modifying any information given by the signatory, and we gain a factor of almost 20 for the verification part while also heavily reducing its memory cost.

Chapter 6

Tighter Bounds: Accelerating NewHope

The work here is the product of a short-term collaboration with Dr Vincent Zucca when he visited us for a post-doc. The spirit of this work is very different from the previous chapters where novel structures and "unpopular" algorithms (or approaches) were explored, however as it is also a contribution on post-quantum cryptography we believe it should also be part of this thesis.

This work is a short[a] study on the New Hope protocol which is one of the candidates for the NIST PQC competition.

Initially, we submitted it for CHES 2019 but we had to modify our work to match the most recent version of the specifications given in the NIST website which we were unaware of: the specifications were updated few weeks before our submission and to our regret, we lacked attention at that time. Furthermore, the work of [ZXZ+18] came to our attention thanks to the reviewers comments, which we were also unaware of but was clearly related to our work. Meanwhile, the work of [ZPL19] appeared online after a few weeks and we also need to take their contribution into consideration.

We would like to thank the reviewers for CHES 2019 as their comments were overall very helpful.

All of this led to this present chapter, which is mostly a paraphrasing of the last

[a] time-consuming as implementation and testing were concerned, but *short* on what we could write. Nevertheless, it allowed me to deepen my knowledge about AVX (Advanced Vector Extensions) implementations so it is not "wasted" time on a personal level. However, our sentiment publication-wise was that the work produced was "not worth the effort".

version of the paper we submitted. At the moment of writing this work should still be under review. However we expect it to be published eventually.

6.1 Introduction

The NewHope protocol [ADPS16b] is a KEM whose security is based on the RLWE problem and has been reported by the NIST as competitive in term of bandwidth and clock-cycles despite quite conservative parameters[b].

It allows two users to exchange a common 256 bits key in order to secure their future communications. However the small errors ensuring the security of the protocol can add together and make it fail, i.e. the two users will not have the same key at the end. In order to ensure this only happens with very low probability, the modulus q must be chosen to be large enough in comparison to the size of the errors. In [ADPS16b] the authors have shown that the modulus $q = 12289$ was large enough for this purpose. This modulus has not been chosen randomly since it is the smallest one allowing to use the NTT algorithm in dimension $n = 512$ or $n = 1024$, which permits to compute efficiently multiplications of elements. Considering that the multiplication is the bottleneck of the arithmetic in this scheme, and that vectorized implementations of NTT are the most efficient way to perform this task on these dimensions ([Sei18]), it is crucial to ensure good performance.

In this work, we review the probability error analysis made in [ADPS16b], also used in the slightly different version [ADPS16a] submitted to the NIST, and show that the actual bound is much lower than what was initially evaluated. This allows to justify the use of a smaller modulus q while maintaining a very low probability failure. Reducing the size of the modulus benefits directly to the bandwidth requirements since elements will be represented on less bits, and to the security of the scheme which is increased. However, given that the original modulus was the smallest one allowing the use of the NTT in dimensions 512 and 1024, one needs to find an alternative strategy to compute the product of elements efficiently. We propose different alternative moduli and, similarly to previous works ([Moe76, ZXZ+18, LS19, ZPL19]), we show that by combining NTTs of smaller dimensions with other multiplication algorithms we obtain competitive performance with the state-of-the-art. More precisely, two of these moduli increase both the compactness and the performance up to 15% without reducing significantly the level of security (−1% in the worst-case). The detailed parameters can be found in Table 6.4. Note that our implementations, sequential and vectorized, work in constant time and are based on Seiler's work ([Sei18]) which is, to the best of our knowledge,

[b] https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

the current state-of-the-art.

6.2 Related Art

Optimising the efficiency of the submissions to the NIST is a hot topic. In a certain sense, `Kyber` ([BDK+18]), which is essentially the adaptation of `NewHope` to the Module-LWE setting, has been created to achieve the best possible performance from `NewHope`. However the parameters of `NewHope` remain, even now, very conservative. It has been mentionned recently that one could reduce the modulus size of `NewHope` to improve the compactness of the scheme ([ZXZ+18]) while performing the arithmetic with a classical trick: combining a naive multiplication algorithm with smaller NTTs ([Moe76]). This work has been recently improved by using a Karatsuba pattern instead of the naive algorithm ([ZPL19]). However since the authors use the same failure probability analysis than in [ADPS16b], the parameters they have derived are far from optimal. Moreover the design of their arithmetic techniques has lead to slightly worst performance than in the original setting.

In the case of [ZXZ+18], the design of their arithmetic technics has lead to slightly worst performance than in the original setting. Moreover, since these works rely on the same failure probability analysis than [ADPS16a], the parameters derived by their authors are not optimal.

Using the idea of [ZXZ+18], Lyubaschevsky and Seiler have proposed a different arithmetic design. The idea is to perform an uncomplete – i.e. when the defining polynomial does not factorize in linear terms, NTT first and then perform the products modulo the small, not linear, factors ([LS19]). This has lead to significantly better performance and has been included in `Kyber`'s implementation since then ([RJL+19]).

`NewHope` implementation of the NTT ([ADPS16b]) which smartly combines Montgomery reductions ([Mon85]) and floating point operations, was considered, to the best of our knowledge, as the reference implementation at the time and was also used in `Kyber`. However in a remarkable work ([Sei18]), Seiler showed that by using a slightly different Montgomery reduction algorithm, and vectorized integer instructions for the NTT, one could gain a factor 4 (resp. 6) over the polynomial multiplication of `NewHope` (resp. `Kyber`). His work has been included in `Kyber`'s implementation recently ([RJL+19]).

6.3 Notations for this chapter

Let m be a power of two, and let $\mathbf{\Phi}_m(X) = X^n + 1$ be the cyclotomic polynomial of index m with $n = \varphi(m) = m/2$ with φ the Euler's totient function. The associated cyclotomic ring will be denoted $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ and for $q \geq 1$ we will denote the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathbb{R} \cong (\mathbb{Z}/q\mathbb{Z})[X]/(X^n + 1)$. Bold lower-case letters $\boldsymbol{a}$ will denote elements in $\mathcal{R}$ which can be seen as polynomials of degree strictly smaller than n with integer coefficients.

The notation $[\cdot]_q$ will represent the centred reduction of an integer modulo q in $[-q/2, q/2)$ and can be extended to polynomials by applying it coefficient-wise. The flooring (resp. the rounding to the nearest integer) of a rational number will be denoted $\lfloor\cdot\rfloor$ (resp. $\lfloor\cdot\rceil$) and these notations will be used for polynomials similarly.

ψ_k will denote a centered binomial distribution of parameter k (elements sampled in $[-k; k] \cap \mathbb{Z}$). Sampling from ψ_k can be done by computing $\sum_{i=1}^{k}(b_i - b'_i)$ where b_i and $b'_i \in \{0, 1\}$ are independent uniformly sampled bits. ψ_k^n will denote the distribution over $\mathcal{R}$ where all the coefficients of a polynomial are sampled independently from ψ_k. Finally, for a distribution $\mathcal{D}$ (resp. a set $\mathcal{I}$), $a \overset{\$}{\leftarrow} \mathcal{D}$ (resp. $a \overset{\$}{\leftarrow} \mathcal{I}$) will mean that a is sampled randomly from $\mathcal{D}$ (resp. uniformly in $\mathcal{I}$).

6.3.1 `NewHope`

For the sake of completeness, `NewHope` protocol is recalled in Figure 6.1. We also present and discuss the main points of the protocol to ease our next analysis. For further details concerning the protocol the interested reader can refer to [ADPS16b].

Alice (server)		**Bob (client)**
$s \xleftarrow{\$} \{0,\dots,255\}^{32}$		
$\boldsymbol{a} \xleftarrow{\$} \text{Parse}(\texttt{SHAKE128}(s))$		
$\boldsymbol{s}_1, \boldsymbol{e}_1 \xleftarrow{\$} \psi_k^n$		
$\boldsymbol{b} \leftarrow [\boldsymbol{a} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1]_q$		$\boldsymbol{s}_2, \boldsymbol{e}_2, \boldsymbol{e}_3 \xleftarrow{\$} \psi_k^n$
$m_a \leftarrow \text{encodeA}(s, \boldsymbol{b})$	$\xrightarrow{m_a}$	$(s, \boldsymbol{b}) \leftarrow \text{decodeA}(m_a)$
		$\boldsymbol{a} \xleftarrow{\$} \text{Parse}(\texttt{SHAKE128}(s))$
		$\boldsymbol{u} \leftarrow [\boldsymbol{a} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2]_q$
		$\nu \leftarrow \{0,1\}^{256}$
		$\nu' \leftarrow \texttt{SHA3-256}(\nu)$
		$\boldsymbol{\kappa} \leftarrow \texttt{NHSEncode}(\nu')$
		$\boldsymbol{c} \leftarrow [\boldsymbol{b} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_3 + \boldsymbol{\kappa}]_q$
		$\hat{\boldsymbol{c}} \leftarrow \texttt{NHSCompress}(\boldsymbol{c})$
$(\boldsymbol{u}, \hat{\boldsymbol{c}}) \leftarrow \text{decodeB}(m_b)$	$\xleftarrow{m_b}$	$m_b \leftarrow \text{encodeB}(\boldsymbol{u}, \hat{\boldsymbol{c}})$
$\boldsymbol{c}' \leftarrow \texttt{NHSDecompress}(\hat{\boldsymbol{c}})$		$\mu \leftarrow \text{SHA3-256}(\nu')$
$\boldsymbol{\kappa}' = [\boldsymbol{c}' - \boldsymbol{u} \cdot \boldsymbol{s}_1]_q$		
$\nu' \leftarrow \texttt{NHSDecode}(\boldsymbol{\kappa}')$		
$\mu \leftarrow \texttt{SHA3-256}(\nu')$		

Figure 6.1: `NewHope` Protocol (without reconciliation) [ADPS16a]

Encoding and compression functions

As shown in Figure 6.1, the key $\nu' \in \{0,1\}^{256}$ is encoded by Bob as an element $\boldsymbol{\kappa} \in \mathcal{R}_q$ through an encoding function `NHSEncode` and Alice recovers it through `NHSDecode`. Additionally, a compression function `NHSCompress` is used in order to reduce the size of the elements sent by Bob to Alice. The way these functions work is detailed below.

Let $\boldsymbol{c} \in \mathcal{R}_q$ and $t \ll q$ be the compression parameter, the compression function consists in keeping only the $\log_2 t$ most significant bits of each coefficient, thus:

$$\texttt{NHSCompress}(\boldsymbol{c}) = \hat{\boldsymbol{c}} \in \mathcal{R}_t$$

$$\text{with } \hat{c}_i = \lfloor c_i \cdot t/q \rceil \bmod t \text{ for } i \in \{0,\dots,n-1\}.$$

The decompression function consists in applying the reverse transformation:

$$\texttt{NHSDecompress}(\hat{\boldsymbol{c}}) = \boldsymbol{c}' \in \mathcal{R}_q$$

$$\text{with } c'_i = \lfloor \hat{c}_i \cdot q/t \rceil \text{ for } i \in \{0,\dots,n-1\}$$

which allows to recover the original polynomial with an error $\boldsymbol{\epsilon}_{comp} \in \mathbb{R}$, such that $||\boldsymbol{\epsilon}_{comp}||_\infty < q/2t + 1/2$, due to the successive roundings.

The 256-bit key ν' is encoded in $\mathcal{R}_q$ thanks to a repetition code which encodes each bit of ν' in $n/256$ coefficients of $\boldsymbol{\kappa}$:

$$\begin{array}{rccc} \texttt{NHSEncode}: & \{0;1\}^{256} & \longrightarrow & \mathcal{R}_q \\ & \nu' & \mapsto & \boldsymbol{\kappa} = \sum\limits_{i=0}^{n-1} \kappa_i \cdot X^i \end{array}$$

with $\kappa_i = \nu'_i \cdot \lfloor q/2 \rfloor$ for $0 \leq i \leq 256$ and $\kappa_{i+256\cdot j} = \kappa_i$ for $0 \leq j < n/256$.

Once the message decompressed, Alice has to extract the key ν' from $\boldsymbol{\kappa}'$ which is equal to the following modulo q:

$$\begin{aligned} \boldsymbol{\kappa}' &= \boldsymbol{c} + \boldsymbol{\epsilon}_{comp} - (\boldsymbol{a} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2) \cdot \boldsymbol{s}_1 \\ &= (\boldsymbol{a} \cdot \boldsymbol{s}_1 + \boldsymbol{e}_1) \cdot \boldsymbol{s}_2 + \boldsymbol{e}_3 + \boldsymbol{\kappa} + \boldsymbol{\epsilon}_{comp} - (\boldsymbol{a} \cdot \boldsymbol{s}_2 + \boldsymbol{e}_2) \cdot \boldsymbol{s}_1 \\ &= \texttt{NHSEncode}(\nu') + \boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1 + \boldsymbol{e}_3 + \boldsymbol{\epsilon}_{comp} \end{aligned}$$

Thus $\boldsymbol{\kappa}'$ corresponds to the encoding of the original key ν' plus some error terms. Therefore, if the error term is not too large, the key ν' can be extracted from $\boldsymbol{\kappa}'$ through the following decoding function:

$$\begin{array}{rccc} \texttt{NHSDecode}: & \mathcal{R}_q & \longrightarrow & \{0;1\}^{256} \\ & \boldsymbol{\kappa}' & \mapsto & \nu' \end{array}$$

where for each $0 \leq i < 256$, ν'_i is recovered by checking the size of the difference between the coefficients of $\boldsymbol{\kappa}'$ and $\lfloor q/2 \rfloor$, which is the value of the coefficients encoding $\nu'_i = 1$. More precisely we check whether:

$$\sum_{j=0}^{n/256-1} |[\kappa'_{i+256\cdot j}]_q - \lfloor q/2 \rfloor|$$

is smaller than $(n/256 \cdot \lfloor q/2 \rfloor)/2$ ($\nu'_i = 1$) or not ($\nu'_i = 0$). Indeed without the error terms (i.e. if $\boldsymbol{\kappa}' = \mathrm{NHSEncode}(\nu')$) then when $\nu'_i = 1$ (resp. 0), the sum would be equal to 0 (resp. $n/256 \cdot \lfloor q/2 \rfloor$). Therefore to ensure the success of the decoding we must ensure that the error remains smaller, in absolute value, than the median value $B_{\mathrm{dec}} = (n/256 \cdot \lfloor q/2 \rfloor)/2$ with very high probability. This comes to ensure that for any $0 \leq i < 256$:

$$\sum_{j=0}^{n/256-1} |(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1 + \boldsymbol{e}_3 + \boldsymbol{\epsilon}_{comp})_{i+256\cdot j}| < B_{\mathrm{dec}} \tag{6.1}$$

A word about the parameters

In order to be able to encode a 256 bits key, n must be at least 256. In their submission package[c], the authors propose two instantiations with different dimensions: $n = 512$ and $n = 1024$ denoted as `NewHope512` and `NewHope1024`, respectively. Even though they recommend to use `NewHope1024` for security reasons, they show that `NewHope512` still ensure a descent security while offering performance roughly twice faster.

Because the bottleneck of the arithmetic in $\mathcal{R}_q$ is by far the multiplication of elements, the usage of the efficient NTT is essential to ensure good performance. Hence, while the protocol can be formally defined for any cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$, taking m different from a power-of-two would prevent to use the NTT and would complicate the generation of the error ([LPR13]). In order to ensure the existence of the roots of unity required for the NTT, q must be congruent to 1 modulo $2n$. Therefore it was set to the smallest prime satisfying this condition which is $q = 12289$ for both $n = 512$ and $n = 1024$.

Finally, for efficiency reasons, the error distribution has been chosen as a centered binomial distribution, instead of a discrete Gaussian as it is usually the case for the (Ring)-LWE problem. Both `NewHope512` and `NewHope1024`, use $k = 8$ as parameter for the binomial distribution. Finally, the compression parameter is chosen as $t = 8$ for both versions.

6.3.2 Number Theoretic Transform (NTT)

Choosing q prime and such that $q \equiv 1 \bmod 2n$ ensures the existence of a primitive $2n$-th roots of unity modulo q that we will denote ζ. Since the odd powers ζ, are also the roots of $X^n + 1$ modulo q, the polynomial factors in linear terms over $\mathbb{Z}_q$. Therefore the CRT gives us the ring isomorphism:

$$\begin{array}{rcc} \mathcal{R}_q & \rightarrow & \mathbb{Z}_q[X]/(X-\zeta) \times \cdots \mathbb{Z}_q[X]/(X-\zeta^{2n-1}) \\ \boldsymbol{a} & \mapsto & (\boldsymbol{a}(\zeta),\ \boldsymbol{a}(\zeta^3), \ldots,\ \boldsymbol{a}(\zeta^{2n-1})) \end{array}$$

The NTT (resp. inverse NTT) allows to compute this morphism (resp. its inverse) in quasi-linear time, through an FFT algorithm. Once in NTT form additions and multiplications can be performed coefficient-wise, and thus in linear time. Therefore the product of elements $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}_q$ can be performed by computing $\boldsymbol{c} = \mathtt{invNTT}(\mathtt{NTT}(\boldsymbol{a}) \odot \mathtt{NTT}(\boldsymbol{a}))$, where $\odot$ denotes the product coefficient-wise. Overall a product can thus be computed in quasi-linear time ($O(n \log n)$). Efficient ways of

[c] https://newhopecrypto.org/resources.shtml

implementating NTTs and its inverse on a vectorized implementation can be found in [Sei18].

6.4 Analysis of the failure probability

In this section we review the failure probability made in [ADPS16b] and refine it. In a nutshell, we will follow their proof but instead of bounding the probability twice using the Chernoff-Cramer inequality and a lemma about subgaussian variables, we will bound it only once through Chernoff-Cramer inequality and deduce the rest from numerical simulations.

6.4.1 About key-encoding and multiplication

We consider the cyclotomic polynomials $\boldsymbol{r}(X) = X^n + 1$ and $\boldsymbol{s}(X) = X^{n/256} + 1$, in particular we have $\boldsymbol{r}(X) = \boldsymbol{s}(X^{256})$. The associated polynomial rings are denoted $\mathcal{R} = \mathbb{Z}[X]/(\boldsymbol{r})$ and $\mathcal{S} = \mathbb{Z}[X]/(\boldsymbol{s})$.

For any $\boldsymbol{a} = a_0 + a_1 \cdot X \cdots + a_{n-1} \cdot X^{n-1} \in \mathbb{R}$ and $0 \leq \boldsymbol{i} \leq 255$, $\boldsymbol{a}'_i(X) \in \mathcal{S}$ will denote:

$$\begin{array}{ll} a_i + a_{i+256} \cdot X & \text{if } n = 512; \\ a_i + a_{i+256} \cdot X + a_{i+512} \cdot X^2 + a_{i+768} \cdot X^3 & \text{if } n = 1024. \end{array}$$

Hence, any $\boldsymbol{a} \in \mathbb{R}$ can be decomposed in the following way:

$$\boldsymbol{a}(X) = \sum_{i=0}^{255} \boldsymbol{a}'_i(X^{256}) \cdot X^i.$$

This decomposition is somehow preserved by the product in $\mathcal{R}$, since for any $\boldsymbol{a}, \boldsymbol{b}$ in $\mathcal{R}$ and $0 \leq \boldsymbol{i} < 256$ we have:

$$(\boldsymbol{a} \cdot \boldsymbol{b})'_i = \left[\sum_{j=0}^{i} (\boldsymbol{a}'_j \cdot \boldsymbol{b}'_{i-j}) + \sum_{j=i+1}^{255} \pi_{\mathcal{S}}(\boldsymbol{a}'_j \cdot \boldsymbol{b}'_{256+i-j}) \right] \in \mathcal{S}$$

where $\pi_{\mathcal{S}}$ denotes the cyclic shift over $\mathcal{S}$ given by the multiplication by X corresponding to:

$$\begin{array}{ll} \pi_{\mathcal{S}}(\boldsymbol{a}') = -a'_1 + a'_0 \cdot X & \text{if } n = 512; \\ \pi_{\mathcal{S}}(\boldsymbol{a}') = -a'_3 + a'_0 \cdot X + a'_1 \cdot X^2 + a'_2 \cdot X^3 & \text{if } n = 1024. \end{array}$$

If we assume that the coefficients of $\boldsymbol{a}$ and $\boldsymbol{b}$ are sampled independently then for each $\boldsymbol{i}$, $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ is a sum of 256 independent polynomials of $\mathcal{S}$:

- the $\boldsymbol{i} + 1$ first polynomials follow the distribution of a polynomial product in $\mathcal{S}$;

- the $255 - i$ last follow the distribution of a shifted polynomial product in $\mathcal{S}$, through $\pi(\mathcal{S})$.

6.4.2 Distribution of the key-encoding error

For any $\boldsymbol{a'}$ and $\boldsymbol{b'}$ in $\mathcal{S}$ we have:

$$\begin{aligned}&\text{if } n = 512\\ (\boldsymbol{a'} \cdot \boldsymbol{b'})(X) = \ & (a'_0 b'_0 - a'_1 b'_1) + (a'_0 b'_1 + a'_1 b'_0) \cdot X\end{aligned}$$

$$\begin{aligned}&\text{if } n = 1024\\ (\boldsymbol{a'} \cdot \boldsymbol{b'})(X) = \ & \ \ (a'_0 b'_0 - a'_1 b'_3 - a'_2 b'_2 - a'_3 b'_1)\\ &+ (a'_0 b'_1 + a'_1 b'_0 - a'_2 b'_3 - a'_3 b'_2) \cdot X\\ &+ (a'_0 b'_2 + a'_1 b'_1 + a'_2 b'_0 - a'_3 b'_3) \cdot X^2\\ &+ (a'_0 b'_3 + a'_1 b'_2 + a'_2 b'_1 + a'_3 b'_0) \cdot X^3\end{aligned}$$

From now, we will denote χ_k the distribution of $\boldsymbol{a'} \cdot \boldsymbol{b'} \in \mathcal{S}$ where $\boldsymbol{a'}$ and $\boldsymbol{b'}$ are sampled from $\psi_k^{n/256}$ independently. From the above expression it is straightforward to notice that χ_k is symmetric because $\psi_k^{n/256}$ is. Moreover, since $\psi_k^{n/256}$ is a centered distribution it is invariant under $\pi_{\mathcal{S}}$ and because $\pi_{\mathcal{S}}(\boldsymbol{a'} \cdot \boldsymbol{b'}) = \boldsymbol{a'} \cdot \pi_{\mathcal{S}}(\boldsymbol{b'})$, χ_k is also invariant under $\pi_{\mathcal{S}}$. Therefore if $\boldsymbol{a}, \boldsymbol{b} \overset{\$}{\leftarrow} \psi_k^n$ then for any $0 \leq i < 256$ $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ can be seen as a sum of 256 independent random variables following the distribution χ_k.

6.4.3 Estimation of the probability failure

In order to estimate the probability than the decryption fails we will need to have an estimation on the tail concentration of a sum of independent identically distributed (i.i.d.) random variables. Similarly to [ADPS16b] we use Chernoff bound for this:

Theorem 8 (Chernoff bound)**.** *Let $\mathcal{D}$ be a distribution over $\mathbb{R}$ and X be a sum of ℓ i.i.d. random variables $X_1, \ldots, X_\ell$ of law $\mathcal{D}$, then for any $t > 0$ such that $\mathbb{E}[e^{t(X_i)}] < +\infty$ and for any $a \in \mathbb{R}$ it holds that:*

$$\mathbb{P}(X \geq a) \leq \exp(-ta + \ell \ln(\mathbb{E}[e^{tX_i}])).$$

Now, in order to ensure (6.1), and thus the success of the decapsulation, with a certain probability we need to have:

$$\|(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i + (\boldsymbol{e}_3)'_i + (\boldsymbol{\epsilon}_{comp})'_i\|_1 < B_{\text{dec}}$$

for all $0 \leq i < 256$. Since $\boldsymbol{e}_3 \overset{\$}{\leftarrow} \psi_k^n$ and because each coefficients of $\boldsymbol{\epsilon}_{comp}$ follows a law almost uniform over $[-q/2t, q/2t]$, the only difficulty comes from the the evaluation of $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$. Similarly to [ADPS16b], we use the fact that for any real vector $\boldsymbol{x} \in \mathbb{R}^d$ we have:

$$\|\boldsymbol{x}\|_1 = \max_{\boldsymbol{y} \in \{\pm 1\}^d} \langle \boldsymbol{x} \ , \ \boldsymbol{y} \rangle \, .$$

As explained in Section 6.4.1, for any $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^2$ each $(\boldsymbol{a} \cdot \boldsymbol{b})'_i$ is the sum of 256 products of elements $\boldsymbol{a}'_j \cdot \boldsymbol{b}'_j \in \mathcal{S}$ and thus $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i = (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ is the sum of 512 such products. Seeing an element of $\mathcal{S}$ as a vector of $\mathbb{Z}^{n/256}$ it holds that:

$$\|(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i\|_1 = \max_{\boldsymbol{y} \in \{\pm 1\}^{n/256}} \left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i \ , \ \boldsymbol{y} \right\rangle \tag{6.2}$$

where $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i$ and $(\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ can both be seen as a sum of 256 independent random variables following the law χ_k (see Section 6.4.1). As a consequence, $(\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i$ can be seen as a sum of 512 i.i.d. random variables. Moreover, remark that because of the different symmetries in χ_k, if X is a random variable of law χ_k, and $\boldsymbol{y} \in \{\pm 1\}^{n/256}$ then the distribution of $\langle X \ , \ \boldsymbol{y} \rangle$ is independent of the choice of $\boldsymbol{y}$. Therefore, because of the linearity of the scalar product $\left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2)'_i - (\boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i \ , \ \boldsymbol{y} \right\rangle$ can be seen as a sum of 512 independent random variables of the form: $\langle X \ , \ \boldsymbol{y} \rangle$ where $X \overset{\$}{\leftarrow} \chi_k$.

We have simulated the law of such a scalar product in rational arithmetic using the `GMP`[d] library ([Gt]) in a C++ script. Note that because of the size of the support of this law ($(2k+1)^{n/128}$), this takes quite a long time to simulate (around 70 min on a laptop with an intel i7-4810MQ@2.80GHz for $n = 1024$ and $k = 8$). This is not much of a problem since the law needs only to be computed once and can then be saved.

Once this law and the law of $\boldsymbol{\epsilon}_{comp}$ simulated, we can use Chernoff bound to get:

$$\mathbb{P}\left(\left\langle (\boldsymbol{e}_1 \cdot \boldsymbol{s}_2 - \boldsymbol{e}_2 \cdot \boldsymbol{s}_1)'_i + (\boldsymbol{e}_3)'_i + (\boldsymbol{\epsilon}_{comp})'_i \ , \ \boldsymbol{y} \right\rangle \geq B_{\text{dec}}\right)$$

$$\leq \exp\left(-ta + 512 \ln \mathbb{E}(e^{tX}) + \tfrac{n}{256}\left(\ln \mathbb{E}(e^{tY}) + \ln \mathbb{E}(e^{tZ})\right)\right)$$

where $X \overset{\$}{\leftarrow} \langle \chi_k \ , \ y \rangle$, $Y \overset{\$}{\leftarrow} \psi_k$ and Z follow the law of $\boldsymbol{\epsilon}_{comp}$.

From there, similarly to [ADPS16b], we have deduced a bound on the probability that the decapsulation fails by union-bounding over the $2^{n/256}$ choices of $\boldsymbol{y}$ and the

[d] https://gmplib.org/

256 possible i. We have computed this bound on the probability using the `MPFR`[e] library ([FHL+07]) for the floating point arithmetic with a precision set to 2048 bits. Our script for computing the probability is available at: `https://gitlab.com/zuccav/newhope-decryption-failure-probability`.

Table 6.1 summarizes the results we have obtained and compare them to those of `NewHope` as they are given in its specifications ([ERJ+18]). As one can see, the probability we obtain is way smaller than the original evaluation which means that one can decrease the size of the modulus while keeping a very small probability of failure.

	Decapsulation error probability	
	`NewHope512`	`NewHope1024`
Original analysis ([ERJ+18])	2^{-213}	2^{-216}
Our analysis (using Chernoff bound)	2^{-393}	2^{-412}

Table 6.1: Decryption Failure Probability of New Hope with binomial parameter $k = 8$ and $q = 12289$.

As for `NewHope` and other related works such as [Saa17], our analysis makes implicitly the assumption that each term are independents when union-bounding over the 256 possibles i. This is clearly not true since they all depend on all the coefficients of the $\boldsymbol{e}_i$s and the $\boldsymbol{s}_j$s. Therefore, while the distribution of each of the terms is the same and precisely analysable, they are not fully independent. In [DVV19], the authors show that this independency hypothesis leads to underestimations of the failure probability when using an error correcting code, but seems to suit constructions not using one. Althouh `NewHope` uses a repetition code, in our case we have simulated exactly the law involving the $n/256$ coefficents correlated via the repetition code (law of $\langle X\ ,\ \boldsymbol{y}\rangle$ for $X \overset{\$}{\leftarrow} \chi_k$), hence the hypothesis is only applied on the 256 polynomials of $\boldsymbol{\mathcal{S}}$ which do not interfer with each other via the code. Therefore, according to [DVV19], we can expect our bounds to be tight.

Note that one could use (6.4.1) and completely simulate the final law by computing 256 convolutions instead of using Chernoff bound. This would give the exact failure probability, like in `Kyber`, and not only an upper bound. However this method is not as efficient in our case. Indeed since `Kyber` has a different reconciliation mechanism, they only need to compute the convolutions of 256 random variables $X_i = \boldsymbol{e} \cdot \boldsymbol{s}$, where $\boldsymbol{e}, s \overset{\$}{\leftarrow} \Psi_k$, whose law support is of size 7 ($k = 2$ in their case). For `NewHope` we have to compute the convolutions of 512 random variables following the law of $\langle X\ ,\ \boldsymbol{y}\rangle$ whose support is of size 65 for $k = 2$ and $n = 1024$. This takes around 1h20 in `C++` on

[e] https://www.mpfr.org

the aforementionned laptop against 0.5 seconds using Chernoff bound and against a few seconds in `python` for `Kyber`. In our experiments we have only observed a difference up to 5 bits between the two variants, hence we rather used Chernoff bound when computing failure probabilities for $k \geq 3$.

6.5 Multiplication strategies for smaller moduli

The lower failure probability of `NewHope` leaves some extra-freedom to optimize further the parameters of the scheme. There are essentially two options: either increase the parameter k and increase the security of the scheme, or reduce the size of the modulus q to make it more compact and, possibly, to improve its performance. With the first option one can use up to $k = 12$ and maintain the failure probability lower than 2^{-190} and 2^{-200} for $n = 512$ and 1024 respectively. In this case, the security of the scheme is increased from 101 (resp. 233) to 107 (resp. 246) bits of post-quantum security[f]. We investigate the second option in the next sections.

6.5.1 Choose the moduli

Using a smaller modulus than $q = 12289$ does not allow to use a full NTT for the dimensions of `NewHope`. However one can still use a modulus allowing to use NTTs of smaller sizes, for instance Kyber modulus $q = 7681$ allows to perform NTTs of size 256. Therefore the idea would be to mix small NTT's with other multiplication algorithms. Since NTT's are the most efficient way to perform polynomial products in high dimensions, one needs to choose moduli with as many NTT levels as possible – i.e. the largest power-of-two n such that $q \equiv 1 \bmod 2n$. Table 6.2 presents possible moduli with the number of NTT levels they allow to perform.

q	12289	7681	3329	2017	1601	1409	769
L	10	8	7	4	5	6	7

Table 6.2: Possible moduli

The security of a RLWE instantiation essentially depends on the ratio between the modulus q and the standard deviation of the error distribution σ_{err}. Hence a small modulus allows to increase the security and also to reduce the size of the elements and thus the communication costs. However it will increase the failure probability, so one might have to use a smaller k. Finally the efficiency of the implementation

[f] using the script `PQsecurity.py` of the NIST submission package: https://newhopecrypto.org/resources.shtml

directly depends on the number of NTT levels available. Therefore one might consider different moduli given the considered trade-off.

In [ZXZ⁺18] and [ZPL19] the authors proposed to use q as small as 3329, however since they relied on the original probability analysis they could hardly justify to use smaller moduli. As a consequence the binomial and compression parameters they proposed were not optimal.

6.5.2 Mutiplication strategies

The idea of mixing classical multiplication algorithms with FFT is well-known and was already studied in [Moe76]. The idea is to split the polynomials $\boldsymbol{a}$ and $\boldsymbol{b}$ into smaller ones and perform the product of the small polynomials with the small NTTs before reconstructing the whole product. The splitting is done between the even and the odd coefficients of our polynomials. More precisely we decompose $\boldsymbol{a}$ as:

$$\begin{aligned}\boldsymbol{a}(X) = \sum_{i=0}^{n-1} a_i \cdot X^i &= \sum_{i=0}^{n/2-1} a_{2i} \cdot X^{2i} + \sum_{i=0}^{n/2-1} a_{2i+1} \cdot X^{2i+1} \\ &= \sum_{i=0}^{n/2-1} a_{2i} \cdot Y^i + X \sum_{i=0}^{n/2-1} a_{2i+1} \cdot Y^i \\ &= \boldsymbol{a}_0(Y) + \boldsymbol{a}_1(Y) \cdot X \text{ with } Y = X^2\end{aligned}$$

with $\boldsymbol{a}_0(Y)$ and $\boldsymbol{a}_1(Y) \in \mathbb{Z}_q[Y]/(Y^{n/2}+1)$, and $\boldsymbol{b}$ similarly. Thus we can write:

$$\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b} = \underbrace{\boldsymbol{a}_0 \cdot \boldsymbol{b}_0}_{\boldsymbol{c}_0} + \underbrace{(\boldsymbol{a}_0 \cdot \boldsymbol{b}_1 + \boldsymbol{a}_1 \cdot \boldsymbol{b}_0)}_{\boldsymbol{c}_1} X + \underbrace{\boldsymbol{a}_1 \cdot \boldsymbol{b}_1}_{\boldsymbol{c}_2} X^2.$$

where each product can be performed with NTTs of size $n/2$. Thus one can just compute $\tilde{\boldsymbol{a}}_i = \mathtt{NTT}(\boldsymbol{a}_i)$ and $\tilde{\boldsymbol{b}}_i$ for $i = 0, 1$ requiring 4 NTTs of size $n/2$ (instead of 2 NTTs of size n in the original case). Once in the NTT domain additions and products can be computed coefficient-wise.

At this point we obtain 3 polynomials of degree $n/2$ in the NTT domain: $\tilde{\boldsymbol{c}}_0$, $\tilde{\boldsymbol{c}}_1$ and $\tilde{\boldsymbol{c}}_2$. Note that since $X^2 = Y$ one can precompute $\mathtt{NTT}(Y)$ (which corresponds to the $n/2$-th primitive roots of unity) and multiply it to $\tilde{\boldsymbol{c}}_2$ and add the result to $\tilde{\boldsymbol{c}}_1$. Like this one has only to compute two inverse NTTs of size $n/2$ instead of 3 to recover the coefficients of $\boldsymbol{c}$ (instead of 1 inverse NTT of size n in the original case).

$$\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_1 \cdot X + \tilde{\boldsymbol{c}}_2 \cdot X^2 = \underbrace{[\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_2 \odot \mathtt{NTT}(Y)]}_{\mathtt{NTT}(\boldsymbol{c}_0)} + \underbrace{\tilde{\boldsymbol{c}}_1}_{\mathtt{NTT}(\boldsymbol{c}_1)} \cdot X \tag{6.3}$$

Note that this also allows to send a polynomial in NTT representation, as done in `NewHope`, without increasing the communication costs.

The complexity of this approach can be reduced in a straightforward-way by using a subquadratic algorithm such as Karatsuba to reduce the number of multiplications at the price of a few extra-additions [ZPL19]. Of course one can iterate this process on several levels in order to use even smaller NTTs. Note that, although computing 2 NTTs of size $n/2$ is more efficient than computing 1 NTT of size n, the extra operations required to expand and reconstruct the Karatsuba pattern tend to make this approach more costly. As a direct consequence, the efficiency of the approach will decrease with the number of Karatsuba levels used.

Overall this approach consists in using one or several levels of classical multiplications algorithms on the top and perform the small products with the NTTs ([ZXZ+18]). This can be done the other way around by performing big, but uncomplete NTTs on the top as in [LS19]:

$$\mathcal{R}_q \xrightarrow{\cong} \mathbb{Z}_q[X]/(X^2 - \zeta) \times \cdots \mathbb{Z}_q[X]/(X^2 - \zeta^{2n-1}).$$

In this case, since $X^n + 1$ does not factorize in linear terms, the product cannot be performed coefficient wise but modulo the $X^2 - \zeta^i$s instead. Once again this can be iterated on ℓ levels so that one would have to perform product modulo $X^{2^\ell} - \zeta^i$ on the bottom. It is shown in [RJL+19], that this approach can be more efficient than a complete NTT for small values of ℓ ($\ell = 1$ or $\ell = 2$) if correctly implemented. In particular the product modulo the $X^2 - \zeta^i$s can be done very efficiently on a vectorized implementation.

6.5.3 Mixing the strategies

As shown in Table 6.2, we need to perform between 2 ($q = 7681$) and 6 ($q = 2017$) levels of multiplications without NTTs. This means that we have to mutiply polynomials of degree between $2^2-1 = 3$ and $2^6-1 = 63$ with classical algorithms. To do so we are going to mix the two previous strategies by using ℓ_1 levels of Karatsuba on the top and ℓ_2 levels of school-book multiplication on the bottom.

Determining the optimal values of ℓ_1 and ℓ_2 is not straightforward. Indeed, the number of operations one can stack on a 16-bit word without performing modular reduction depends on the size of the modulus. Moreover the cost of the modular reductions and their efficiency depend on the shape of the modulus (see [Sei18]). Additionnally, on a vectorized implementation, the cost of an algorithm cannot be restricted only to the number of operations to perform. One must rather consider the number of registers available, of instructions and loadings to perform, etc.... Therefore, the optimal values of ℓ_1 and ℓ_2 has been determinated experimentally for every moduli by implementing every combination with $0 \leq \ell_1 \leq 4$..

While theoretically more efficient, Karatsuba algorithm only becomes more efficient than school-book multiplications from a certain point. Actually, in our experiments, we have noticed that the cutoff point is around 4, 5 levels of multiplications without NTTs. This means that introducing some level of Karatsuba – i.e. $\ell_1 > 0$ – on our parameters is not interesting when $\ell_1 + \ell_2 \leq 3$.

6.6 Global parameters and experimental results

6.6.1 Security parameters

In practice, `NewHope` security is evaluated through the difficulty to recover the secret $\boldsymbol{s}$ from a RLWE sample $(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e})$. The difficulty of this problem depends on the ratio between the size of the modulus and the standard deviation of the error distribution, in our case $\sqrt{k/2}$, the smaller the ratio, the better the security. Therefore by using a smaller modulus the difficulty of the problem is increased, however not reducing the binomial parameter k would quickly result in a failure probability too large which could lead to serious attacks (see [DGJ$^+$19]).

As a consequence, we have chosen a binomial parameter k as large as possible for the security of the scheme while keeping the failure probability of the protocol around 2^{-200} thus similar to the evaluation of [ERJ$^+$18] over the original parameters. In order to increase the gain over the bandwidth requirement we have also reduced, when possible, the compression parameter t.

Since the conservative analysis of `NewHope` ([ADPS16b]), is currently used in most of the NIST submissions based on lattices. We have computed the evaluated post-quantum bits of security using the same python script[g] than for the original protocol. However the indicated failure probability has been computed with our analysis (see Section 6.4).

[g] available in the NIST submission package: https://newhopecrypto.org/resources.shtml

Eventually for $q = 769$ we have to decrease the binomial parameter to $k = 1$. For such a small k, one must also consider the hybrid attack as presented in [BGPW16] to estimate the complexity of the lattice reduction. We have estimated the cost of this attack by using the `Sage` script available at `https://github.com/lducas/LatRedHybrid`. In our case, these attacks do not seem to impact the global security.

6.6.2 The particular case of $q = 1601$

The modulus $q = 2017$ already fits on 11-bits but only offers 4 levels of NTT leading to poor performance. Hence we have instead selected two other moduli: $q = 1601$ and $q = 1409$ allowing to perform 1 and 2 extra levels of NTT respectively. However choosing $q = 1601$ and $k = 2$ would result in a probability of failure around 2^{-129}, which is far too large and choosing $k = 1$ would have significantly lower the security. Finally increasing the compression parameter t would cancel the benefit over the bandwidth requirement.

Since most of the error size comes from the products $\boldsymbol{s}_1 \cdot \boldsymbol{e}_2 - \boldsymbol{s}_2 \cdot \boldsymbol{e}_1$ and because the RLWE security depends essentially on the size of the error rather than the size of the secret. We have chosen in this cases to use different binomial parameters k_s and k_e for the secrets and the errors respectively. As a consequence, choosing a binary secret – i.e. $k_s = 1$ – and a larger error $k_e = 3$ (resp. $k_e = 2$) increases the security of the scheme while reducing the error probability to 2^{-170} (resp. 2^{-197}) for $q = 1601$ (resp. $q = 1409$). Although for $q = 1601$ this failure probability is far above the 2^{-200} treshold, it is comparable to the failure probability of `Kyber` [RJL+19]. Finally, note that when using a different size of errors and secrets, and more particularly with sparse secrets, one needs to adapt the security evaluation by including weights on the lattice to reduce ([BG14, Alb17]).

6.6.3 Experimental results

We have implemented the different multiplication strategies in `C` with, and without, vectorized (`AVX2`) instructions. We will refer to these implementations as the vectorized and the reference implementation respectively.

Table 6.3 details the impact of the choice of ℓ_1 and ℓ_2 on the performance of the multiplication for four moduli $q = 3329$, 1409, 769 and 1601. The first three moduli are the most interesting ones when considering the gain on the three features: security, compactness and performance. The last one, 1601, shows the impact of ℓ_1 and ℓ_2 when multiplying polynomials of relatively large degree $2^5 - 1 = 31$ ($\ell_1 + \ell_2 = 5$).

Table 6.4 summarizes the different parameters, features and performance of the original version of `NewHope512` and `NewHope1024` together with those of the alternative versions we propose.

In order to estimate the efficiency of our multiplication strategy we have measured the number of cycles required to compute a full product in our reference and vectorized implementations. The ℓ_1 levels of Karatsuba have been implemented iteratively so that we could track the size of the coefficients at any point and only perform the modular reductions when needed.

The values presented in Table 6.3 and Table 6.4 were measured on an average of 2^{20} tests run on a laptop endowed with an Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz using `AVX2` with Turbo Boost and Hyper-Threading turned-off. Our code was compiled with `gcc` version 9.3.1 using the flags: `-O3 -funroll-loops -fomit-frame-pointer -march=native`.

Our experiments confirm the observations of [LS19]: using smaller NTTs on the top with a naive products on the bottom is more efficient than a full NTT with a gain up to 15% (resp. (13%)) for $n = 512$ (resp. $n = 1024$). Because registers in `AVX2` are 256-bit long, they can only handle 16 coefficients on 16 bits words at a time. In this case, having $\ell_2 > 4$ levels is not interesting since one would have to multiply polynomials of degree $2^{2^{\ell_2}} > 16$. Hence it requires two registers to store the coefficients of only one polynomial degrading considerably the performance.

We observe that the most efficient strategies on the reference implementation do not use any level of Karatsuba. On the other hand, on the vectorized implementation mixing a few levels of Karatsuba with the school-book algorithm – i.e. take $\ell_1 > 0$ – can be more efficient when one has more than 4 levels of multiplications to perform without NTTs ($\ell_1 + \ell_2 \geq 4$). As one could expect the speed-ups are more important for the reference implementation than for the vectorized one. This is due to the fact that non-vectorized NTTs are, by far, the bottleneck of the arithmetic. Hence by using several smaller NTTs one gains a lot in performance. However since NTTs are much faster on the vectorized implementation, the gain coming from the use of smaller NTTs is less important and the global speed-ups are thus reduced.

As expected, the modulus $q = 2017$ does not give competitive performance +39% (resp. +77%) for $n = 512$ (resp. $n = 1024$), and that the additional levels of NTT available with $q = 1601$ and $q = 1409$ improves considerably the performance +6.5% (resp. +35%) for $n = 512$ (resp. $n = 1024$) for $q = 1601$ and −9.4% (resp. +4%) for $n = 512$ (resp. $n = 1024$) for $q = 1409$.

Nonetheless most of our parameters allow to obtain competitive performance while gaining around 19% over the bandwidth requirements and up to 10% in security. Moreoever the two moduli 7681, 3329 and 769 do not have any drawback, except a negligible 1% loss of security for 3329. Furthermore the gain of more than 10% on performance comes without having to implement any level of Karatsuba and thus

n	q	$\ell_1 \vert \ell_2$	Clock cycles for one product	
			Reference	Vectorized
512	3329	0|2	**31535**	**3629**
		1|1	34234	3966
		2|0	38478	3936
	1601	0|4	**34663**	**4582**
		1|3	34984	4665
		2|2	38185	4970
	1409	0|3	**29770**	**3849**
		1|2	33469	4189
		2|1	33980	4720
	769	0|2	**31294**	**3543**
		1|1	33668	3985
		2|0	38112	3927
1024	3329	0|3	**64831**	**8377**
		1|2	72335	9092
		2|1	74156	9960
		3|0	80459	9688
	1601	0|5	**83743**	15654
		1|4	91394	13149
		2|3	88483	**12729**
		3|2	106788	13278
		4|1	103737	20706
	1409	0|4	**70637**	9711
		1|3	71740	**9646**
		2|2	79573	10197
		3|1	76516	11723
	769	0|3	**64109**	**8313**
		1|2	71628	8912
		2|1	73261	9848
		3|0	78823	9343

Table 6.3: Impact of ℓ_1 and ℓ_2 on the performance of the multiplication.

leads to a very simple implementation. The modulus $q = 769$ improves significantly the three main features of the protocol: security, compactness and performance while maintaining a failure probability around 2^{-182}, which is comparable to `Kyber` and `SABER`.

Eventually using a smaller modulus q, and a smaller error parameter k, allows to generate significantly less random bits during the execution of the protocol reducing therefore the number of calls to the `SHAKE128` function. Hence, for the whole protocol, we can expect gains in performance beyond the speed-ups presented in Table 6.4 which are only related to the arithmetic.

n	q	k	t	probability of failure	post-quantum bits of security	bandwidth (Bytes)	Reference		Vectorized	
							$\ell_1\|\ell_2$	clock cycles for one product	$\ell_1\|\ell_2$	clock cycles for one product
512	12289	8	8	2^{-393} (C)	101	2016	0|0	57125	0|0	4263
	7681	5	4	2^{-194} (C)	101 (+0%)	1824 (−9.5%)	0|1	35919 (−37%)	0|1	3643 (−15%)
	3329	2	4	2^{-239} (S)	100 (−1%)	1696 (−15.8%)	0|2	31537 (−44%)	0|2	3629 (−15%)
	2017	2	8	2^{-197} (S)	108 (+7%)	1632 (−19%)	0|5	41208 (−26%)	2|3	5948 (+39%)
	1601	1|3	8	2^{-170} (S)	109 (+8%)	1632 (−19%)	0|4	34663 (−39%)	0|4	4582 (+7.5%)
	1409	1|2	8	2^{-197} (S)	107 (+6%)	1632 (−19%)	0|3	29770 (−48%)	0|3	3849 (−9.7%)
	769	1	32	2^{-184} (S)	112 (+10%)	1632 (−19%)	0|2	31294 (−45%)	0|2	3543 (−17%)
1024	12289	8	8	2^{-412} (C)	233	4000	0|0	118461	0|0	9247
	7681	5	4	2^{-208} (C)	233 (+0%)	3616 (−9.6%)	0|2	69503 (−41%)	0|2	8102 (−12%)
	3329	2	4	2^{-249} (S)	230 (−1%)	3360 (−16%)	0|3	64831 (−45%)	0|3	8377 (−9.4%)
	2017	2	8	2^{-204} (S)	245 (+5%)	3232 (−19.2%)	0|6	112066 (−4%)	3|3	16611 (+79%)
	1601	1|3	8	2^{-175} (S)	245 (+5%)	3232 (−19.2%)	0|5	83743 (−29%)	2|3	12729 (+38%)
	1409	1|2	8	2^{-202} (S)	242 (+4%)	3232 (−19.2%)	0|4	70637 (−40%)	1|3	9646 (+4%)
	769	1	32	2^{-182} (S)	250 (+7%)	3232 (−19.2%)	0|3	64109 (−45%)	0|3	8313 (−10%)

Table 6.4: Alternative parameters with the performance of the associated arithmetic for NewHope512 and NewHope1024. (C) and (S) denote whether the probability of failure was computed using Chernoff bound or only simulations respectively.

Summary and conclusion

We refine the probability analysis made in [ADPS16b] and show that we can use tighter parameters in order to increase the security, the compactness and sometimes the performance of NewHope. We propose four alternative set of parameters using smaller moduli. However these moduli do not allow to use the full efficient NTT algorithm in the protocol dimensions. Nonetheless we show that by mixing smaller NTTs with different multiplication algorithms we obtain very competitive performance when compared to the state-of-the-art approach (gain of 16% for some of them). While we are competitive in term of efficiency our alternative parameters allow to increase the security of the protocol up to 8% and reduce the bandwidth requirement by up to 19%. Furthermore some sets of parameters improved the three features of the scheme: compactness, security, performance while the other improve even more the compactness and security but lead to worse performance. For all these sets of parameters we evaluate the failure probability around 2^{-200} as originally evaluated ([ERJ+18]), except for $q = 1601$ where it is as big as 2^{-170}, similarly to Kyber, which should be enough to be protected against attacks such as the one mentionned in [DGJ+19].

Chapter 7

Conclusions and open questions

In this thesis, we presented different structural modifications for lattice-based cryptography. We list here the different conclusions of our work.

7.1 Using lattice intersections to hide a lattice generated by a secret key

In chapter 3, we presented a previously unused structure of lattice-based cryptography and applied it to the reinforcement of **GGH**. As the arising problems are relatively undocumented, further work would be necessary to either exploit that new structure for enhancing other schemes, or extend the existing knowledge in cryptanalysis.

7.2 Using a diagonal dominant matrix to reduce the maximum norm of vectors

In chapter 4, we presented in this chapter the **DRS** scheme, and another method to generate secret keys for providing experimental results on the statistical distribution of the keys generated following Yu and Ducas' attack. We demonstrate that our new approach is sufficient to improve **DRS** to be secure against machine learning attacks as reported earlier in the literature. However, the secret matrix is still diagonal dominant and it remains an open question whether there exists a tight security proof to a well-known problem or if there is any unforeseen weaknesses to diagonal dominant lattices as both Li, Liu, Nitaj and Pan's [LLNP18] and Yu and Ducas's attacks [YD18a] could lead to. The open questions for improvement stated in the original **DRS** report are also still applicable to the last proposed iteration (modifications of the public key and signing algorithm). Overall, we showed that both efficiency and security of such schemes are related to the noise more than the diagonal coefficients. Given a fixed diagonal, [LLNP18, YD18a] showed weakness on particular noise sets. It is unclear if our choice of uniform sampling in the n-dimensional ball is provably secure, but we stress that the literature is very scarce concerning this lattice family and thus lots of open questions remain.

On the technical side, our method to generate random samples is also slow and might need improvement. It also impacts the setup as mentioned earlier, as keeping the current **DRS** parameters one can see the possibility to overflow and go over 64-bits, even though the probability is extremely low, thus changing the public key generation is also left as an open question. The initial **DRS** scheme was very conservative not only on their security but also the manipulated integer size bounds: one might use heuristics to drastically increase the memory efficiency of the scheme and allow some small error probability for example.

7.3 Using Freivalds' algorithm to accelerate signature verifications

In chapter5, we introduced a modification of Freivalds' algorithm to introduce a faster verification method to **DRS**. By introducing a precomputation step that is in the same order of magnitude as the setup in time, we gain a factor of almost 20 for the verification part while also heavily reducing its memory cost. This process is done while not modifying any information given by the signatory. Furthermore, more research should greatly improve this new work.

1. We assumed almost "paranoiac" security requirements, thus a deeper analysis should improve efficiency.

2. We can make use of RNS: stemming from [Gar59] with several applications, finding large arithmetically efficient random groups is exactly what we need.

3. We could generalize to all lattices and **HNF** keys. It needs the extra vector h, but any party can compute h in polynomial time with no security loss.

7.4 A potential improvement on New Hope

In chapter 6, we refine the probability analysis made in [ADPS16b] and show that we can use tighter parameters in order to increase the security, the compactness and sometimes the performance of `NewHope`. We propose four alternative set of parameters using smaller moduli. However these moduli do not allow to use the full efficient

NTT algorithm in the protocol dimensions. Nonetheless we show that by mixing smaller NTTs with different multiplication algorithms we obtain very competitive performance when compared to the state-of-the-art approach (gain of 16% for some of them). While we are competitive in term of efficiency our alternative parameters allow to increase the security of the protocol up to 8% and reduce the bandwidth requirement by up to 19%. Furthermore some sets of parameters improved the three features of the scheme: compactness, security, performance while the other improve even more the compactness and security but lead to worse performance. For all these sets of parameters we evaluate the failure probability around 2^{-200} as originally evaluated ([ERJ+18]), except for $q = 1601$ where it is as big as 2^{-170}, similarly to `Kyber`, which should be enough to be protected against attacks such as the one mentionned in [DGJ+19].

Although [DGJ+19] has shown how decryption failures could be exploited by a malicious adversary, we still do not know exactly which bound should be aimed on the failure probability for a given targeted security in practice. For instance `NewHope` and `Kyber` maintain a similar failure probability regardless of the security level they are aiming for, while `SABER` ([DKSRV18]) adapts its parameters so that the failure probability gets smaller for a higher level of security.

Following the specifications of `NewHope`, we have kept a similar failure probability regardless of the dimension n. However `NewHope512` targets around 100 bits of security and can thus probably tolerate a bigger failure probability than `NewHope1024` which aims at more than 200. Therefore the parameters for $n = 512$ can probably be further optimized. The only question remaining is: what failure probability can we afford in practice?

As another example, we have been conservative on the compression parameter $t = 32$ for $q = 769$ in order to ensure a failure probability as small as possible. Nonetheless, reducing t to 16 would allow to improve further the bandwidth requirement (−22%) at a cost of a failure probability smaller than 2^{-162}. Although this probability is similar to those of `Kyber` and `SABER` ([DKSRV18]) for their most secure parameters (2^{-174} and 2^{-165} respectively), we have chosen to keep $t = 32$ for conservatism.

Bibliography

[ABSS93] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 724–733. IEEE, 1993.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC '97*, pages 284–293. ACM, 1997.

[ADPS16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Newhope without reconciliation. *IACR Cryptology ePrint Archive*, 2016:1157, 2016. `https://eprint.iacr.org/2016/1157`.

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 327–343, 2016.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC '96*, pages 99–108. ACM, 1996.

[Ajt98] Miklós Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998.

[AKS01] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 601–610. ACM, 2001.

[Alb17] Martin R Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in helib and seal. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 103–129. Springer, 2017.

[Ano19] Anonymous. Cards against cryptography. Github, ASIACRYPT 2018, EUROCRYPT 2019, 2018-2019. ”Knapsack Systems, Revisited”, available at `https://github.com/CardsAgainstCryptography/CAC/blob/master/PNGs-to-print/individual-cards/white_FRONT153.png`.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP $\cap$ Co $-$ NP. *J. ACM*, 52(5):749“765, September 2005.

[Bab86] László Babai. On lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Bar86] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 311–323. Springer, 1986.

[BBdV+17] Jens Bauch, Daniel J Bernstein, Henry de Valence, Tanja Lange, and Christine Van Vredendaal. Short generators without quantum computers: the case of multiquadratics. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 27–59. Springer, 2017.

[BCJ11] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 364–385. Springer, 2011.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[BEHZ16] Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 2016.

[BEM16] J-C Bajard, Julien Eynard, and Nabil Merkiche. Multi-fault attack detection for RNS cryptographic architecture. *IEEE 23nd Symposium on Computer Arithmetic*, July 2016.

[BG14] Shi Bai and Steven D Galbraith. Lattice decoding attacks on binary LWE. In *Australasian Conference on Information Security and Privacy*, pages 322–337. Springer, 2014.

[BGJ13] Anja Becker, Nicolas Gama, and Antoine Joux. Solving shortest and closest vector problems: The decomposition approach. *Cryptology ePrint Archive, Report 2013/685*, 2013. `https://eprint.iacr.org/2013/685`.

[BGPW16] Johannes Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In *International Conference on Cryptology in Africa*, pages 24–43. Springer, 2016.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[BI04] J-C Bajard and Laurent Imbert. A full RNS implementation of RSA. *IEEE Transactions on Computers*, 53(6):769–774, 2004.

[BIP04] Jean-Claude Bajard, Laurent Imbert, and Thomas Plantard. Modular number systems: Beyond the mersenne family. In *International Workshop on Selected Areas in Cryptography*, pages 159–169. Springer, 2004.

[BL09] Johannes A Buchmann and Richard Lindner. Density of ideal lattices. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-ZeNTRUm für Informatik, 2009.

[BL18] Daniel J Bernstein and Tanja Lange. Quantum computers: The future attack that breaks today's messages. Macquarie University, 2018. Invited Talk, available at `https://www.mq.edu.au/about/about-the-university/offices-and-units/optus-macquarie-university-cyber-security-hub/news-and-events/news2/news/profs-bernstein-and-lange`.

[Bli14] Hans Frederik Blichfeldt. A new principle in the geometry of numbers, with some applications. *Transactions of the American Mathematical Society*, 15(3):227–235, 1914.

[BLN16] Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. *Dual EC: A Standardized Back Door*, pages 256–281. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. Free access: `https://projectbullrun.org/dual-ec/documents/dual-ec-20150731.pdf`.

[BP04] Jean-Claude Bajard and Thomas Plantard. RNS bases and conversions. In *Optical Science and Technology, the SPIE 49th Annual Meeting*, pages 60–69. International Society for Optics and Photonics, 2004.

[BR91] Richard A Brualdi and Herbert J Ryser. *Combinatorial matrix theory*, volume 39. Cambridge University Press, 1991.

[Bri83] Ernest F Brickell. Are most low-density knapsacks solvable in polynomial time. Technical report, Sandia National Labs., Albuquerque, NM (USA), 1983.

[BS72] Richard H. Bartels and George W Stewart. Solution of the matrix equation ax+ xb= c [f4]. *Communications of the ACM*, 15(9):820–826, 1972.

[BS99] Johannes Blömer and Jean-Pierre Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720. ACM, 1999.

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 369–404. Springer, 2018.

[CAG18] University of Sydney Computational Algebra Group. Magma online. https://magma.maths.usyd.edu.au/calc/, 2018.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 559–585. Springer, 2016.

[Che13] Yuanmi Chen. Lattice reduction and concrete security of fully homomorphic encryption. *Dept. Informatique, ENS, Paris, France, PhD thesis*, 2013.

[CJL+92] Matthijs J Coster, Antoine Joux, Brian A LaMacchia, Andrew M Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. *computational complexity*, 2(2):111–128, 1992.

[CN11] Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.

[Coh93] Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of Graduate Texts in Mathematics. Springer-Verlag, 1993.

[CR88] Benny Chor and Ronald L Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988.

[DFK+20] Dung Hoang Duong, Kazuhide Fukushima, Shinsaku Kiyomoto, Partha Sarathi Roy, Arnaud Sipasseuth, and Willy Susilo. Lattice-based public key encryption with equality test supporting flexible authorization in standard model, 2020. `https://arxiv.org/abs/2005.05308`.

[DFV97] Hervé Daudé, Philippe Flajolet, and Brigitte Vallée. An average-case analysis of the gaussian algorithm for lattice reduction. *Combinatorics, Probability and computing*, 6(4):397–433, 1997.

[DGJ+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In *IACR International Workshop on Public Key Cryptography*, pages 565–598. Springer, 2019.

[Din00] Irit Dinur. Approximating SVP ∞ to within almost-polynomial factors is NP-hard. In Giancarlo Bongiovanni, Rossella Petreschi, and Giorgio Gambosi, editors, *Algorithms and Complexity*, pages 263–276, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[DKSRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2018*, pages 282–305, Cham, 2018. Springer International Publishing.

[DlVP97] Charles-Jean De la Vallée Poussin. *Recherches analytiques sur la théorie des nombres premiers.* Hayez, Imprimeur de l'Académie royale de Belgique, 1897.

[DN12] Léo Ducas and Phong Q Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUsign countermeasures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 433–450. Springer, 2012.

[DP65] NA Derzko and AM Pfeffer. Bounds for the spectral radius of a matrix. *Mathematics of Computation*, 19(89):62–67, 1965.

[DRS$^+$20] Dung Hoang Duong, Partha Sarathi Roy, Willy Susilo, Kazuhide Fukushima, Shinsaku Kiyomoto, and Arnaud Sipasseuth. Lattice-based public key encryption with equality test in standard model, revisited, 2020. `https://arxiv.org/abs/2005.03178`.

[Duc17] L Ducas. Advances on quantum cryptanalysis of ideal lattices. *Nieuw Archief voor Wiskunde*, 5:184–189, 2017.

[Dum18] Jean-Guillaume Dumas. Proof-of-work certificates that can be efficiently computed in the cloud (invited talk). In *International Workshop on Computer Algebra in Scientific Computing*, pages 1–17. Springer, 2018.

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. The impact of error dependencies on ring/mod-LWE /LWR based schemes. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 103–115, Cham, 2019. Springer International Publishing.

[DZ17] Jean-Guillaume Dumas and Vincent Zucca. Prover efficient public verification of dense or sparse/structured matrix-vector multiplication. In *Australasian Conference on Information Security and Privacy*, pages 115–134. Springer, 2017.

[EJ16] Thomas Espitau and Antoine Joux. Adaptive precision LLL and potential-LLL reductions with interval arithmetic. *IACR Cryptology ePrint Archive*, 2016:528, 2016.

[EK40] Paul Erdös and M Kac. The gaussian law of errors in the theory of additive number theoretic functions. *American Journal of Mathematics*, 62(1):738–742, 1940.

[ERJ+18] Alkim Erdem, Avenzi Roberto, Bos Joppe, Ducas Léo, de la Piedra Antonio, Pöppelmann Thomas, Schwabe Peter, and Stebila Douglas. *NewHope Algorithm Specifications and Supporting Documentation*, 1.0 edition, 2018. `https://newhopecrypto.org/data/NewHope_2018_06_14.pdf`.

[eS18] Tomás Oliveira e Silva. Tables of values of pi(x) and of pi2(x). http://sweet.ua.pt/tos/primes.html, 2018.

[FHL+07] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2):13‘‘es, June 2007. `https://www.mpfr.org/`.

[FP85] Ulrich Fincke and Michael Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985.

[FPL] FPLLL team. FPLLL, a lattice reduction library.

[Fre79] Rūsiņš Freivalds. Fast probabilistic algorithms. In *International Symposium on Mathematical Foundations of Computer Science*, pages 57–69. Springer, 1979.

[FS99] Roger Fischlin and Jean-Pierre Seifert. Tensor-based trapdoors for CVP and their application to public key cryptography. In *Cryptography and Coding*, pages 244–257. Springer, 1999.

[FSW14] Felix Fontein, Michael Schneider, and Urs Wagner. Potlll: a polynomial time version of LLL with deep insertions. *Designs, codes and cryptography*, 73(2):355–368, 2014.

[Gar59] Harvey L Garner. The residue number system. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, pages 146–153. ACM, 1959.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540 – 563, 2000.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO'97*, pages 112–131. Springer, 1997.

[GINX16] Nicolas Gama, Malika Izabachene, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: generalized worst-case to average-case reductions and homomorphic cryptosystems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 528–558. Springer, 2016.

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[GLP+12] Filippo Gandino, Fabrizio Lamberti, Gianluca Paravati, Jean-Claude Bajard, and Paolo Montuschi. An algorithmic and architectural study on montgomery exponentiation in RNS. *IEEE Transactions on Computers*, 61(8):1071–1083, 2012.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of hecke points. In *Forum Mathematicum*, volume 15, pages 165–190. Berlin; New York: De Gruyter, c1989-, 2003.

[GMGPG+14] Oscar Garcıa-Morchón, Domingo Gómez-Pérez, Jaime Gutiérrez, Ronald Rietman, Berry Schoenmakers, and Ludo Tolhuizen. HIMMO: A lightweight collusion-resistant key predistribution scheme. online, 2014. `https://eprint.iacr.org/2014/698.pdf`.

[GMRS+15] Oscar Garcia-Morchon, Ronald Rietman, Sahil Sharma, Ludo Tolhuizen, and Jose Luis Torre-Arce. DTLS-HIMMO: Achieving DTLS certificate security with symmetric key overhead. In *European Symposium on Research in Computer Security*, pages 224–242. Springer, 2015.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[GN08] Nicolas Gama and Phong Q Nguyen. Predicting lattice reduction. In *EUROCRYPT 2008*, pages 31–51. Springer, 2008.

[GNR10] Nicolas Gama, Phong Q Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, pages 257–278. Springer, 2010.

[GNVL79] Gene Golub, Stephen Nash, and Charles Van Loan. A hessenberg-schur method for the problem ax+ xb= c. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206. ACM, 2008.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[Gt] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library.*

[Had96] Jacques Hadamard. Sur la distribution des zéros de la fonction ζ(s) et ses conséquences arithmétiques. *Bulletin de la Societé mathematique de France*, 24:199–220, 1896.

[HGJ10] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–256. Springer, 2010.

[HMM98] George Havas, Bohdan S Majewski, and Keith R Matthews. Extended gcd and hermite normal form algorithms via lattice basis reduction. *Experimental Mathematics*, 7(2):125–136, 1998.

[Hou64] Alston S Householder. The theory of matrices in numerical analysis. 1964.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic number theory*, pages 267–288. Springer, 1998.

[HPS$^+$17] Jeff Hoffstein, Jill Pipher, John M Schanck, Joseph H Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRU-encrypt. In *CT-RSA 2017*, pages 3–18. Springer, 2017.

[HW38] G.H. Hardy and E.M. Wright. *An Introduction to THE THEORY OF NUMBERS.* Oxford University Press, London, First Edition 1938.

[Jou93] Antoine Joux. *La réduction des réseaux en cryptographie.* 1993.

[JP06] Marc Joye and Pascal Paillier. Fast generation of prime numbers on portable devices: An update. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 160–173. Springer, 2006.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206. ACM, 1983.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[KB79] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *siam Journal on Computing*, 8(4):499–507, 1979.

[KGP$^+$89] Donald Erwin Knuth, Ronald L Graham, Oren Patashnik, et al. Concrete mathematics. *Adison Wesley*, 1989.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[Kim16] Seungki Kim. On the distribution of lengths of short vectors in a random lattice. *Mathematische Zeitschrift*, 282(3-4):1117–1126, 2016.

[KS93] Tracy Kimbrel and Rakesh Kumar Sinha. A probabilistic algorithm for verifying matrix products using o(n2) time and log2(n)+ o(1) random bits. *Information Processing Letters*, 45(2):107–110, 1993.

[Kun08] Noboru Kunihiro. New definition of density on knapsack cryptosystems. In *International Conference on Cryptology in Africa*, pages 156–173. Springer, 2008.

[KZ73] Aleksandr Korkine and G Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.

[Lee11] Moon Sung Lee. Cryptanalysis of a quadratic compact knapsack public-key cryptosystem. *Computers & Mathematics with Applications*, 62(9):3614–3621, 2011.

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 450–461. Springer, 2006.

[LLNP18] Haoyu Li, Renzhang Liu, Abderrahmane Nitaj, and Yanbin Pan. Cryptanalysis of the randomized version of a lattice-based signature scheme from pkc'08. In *ACISP 2018*, pages 455–466. Springer, 2018.

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO 2009*, pages 577–594. Springer, 2009.

[LMHG13] Cong Ling, Wai Ho Mow, and Nick Howgrave-Graham. Reduced and fixed-complexity variants of the LLL algorithm for communications. *IEEE Transactions on Communications*, 61(3):1040–1050, 2013.

[LO85] Jeffrey C Lagarias and Andrew M Odlyzko. Solving low-density subset sum problems. *Journal of the ACM (JACM)*, 32(1):229–246, 1985.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA 2011*, pages 319–339. Springer, 2011.

[LPMSW19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé, and Alexandre Wallet. An LLL algorithm for module lattices. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 59–90, Cham, 2019. Springer International Publishing. Full version available at `https://eprint.iacr.org/2019/1035.pdf`.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013.

[LPS19] Andrea Lesavourey, Thomas Plantard, and Willy Susilo. On ideal lattices in multicubic fields. 2019. http://nutmic2019.imj-prg.fr/confpapers/MultiCubic.pdf.

[LS19] Vadim Lyubashevsky and Gregor Seiler. Nttru: Truly fast NTRU using ntt. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 180–201, 2019.

[LWXZ11] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. *IACR Cryptology ePrint Archive*, 2011:139, 2011.

[Lyu16] Vadim Lyubashevsky. Future directions in lattice cryptography. PKC, Invited Talk, 2016. http://troll.iis.sinica.edu.tw/pkc16/invited.shtml#vadim.

[Mau95] Ueli M Maurer. Fast generation of prime numbers and secure public-key cryptographic parameters. *Journal of Cryptology*, 8(3):123–155, 1995.

[MG12] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. The Springer International Series in Engineering and Computer Science. Springer US, 2012.

[MH78] Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE transactions on Information Theory*, 24(5):525–530, 1978.

[Mic01] Daniele Micciancio. Improving lattice based cryptosystems using the hermite normal form. In *Cryptography and Lattices*, pages 126–145. Springer, 2001.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *computational complexity*, 16(4):365–411, 2007.

[Mic08] Daniele Micciancio. Efficient reductions among lattice problems. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 84–93. Society for Industrial and Applied Mathematics, 2008.

[Mic10] Daniele Micciancio. Duality in lattice cryptography. Public Key Cryptography, 2010. Invited Talk.

[Min96] Hermann Minkowski. *Geometrie der Zahlen.* B.G. Teubner, Leipzig, 1896.

[Moe76] Robert T Moenck. Practical fast polynomial multiplication. In *Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 136–148. ACM, 1976.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 820–849. Springer, 2016.

[Ngu99] Phong Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from crypto'97. In *CRYPTO'99*, pages 288–304. Springer, 1999.

[Nie90] Valtteri Niemi. A new trapdoor in knapsacks. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 405–411. Springer, 1990.

[NR09] Phong Q Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, 22(2):139–160, 2009.

[NS97] Phong Nguyen and Jacques Stern. Merkle-Hellman revisited: a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In *Annual International Cryptology Conference*, pages 198–212. Springer, 1997.

[NS05] Phong Q Nguyen and Jacques Stern. Adapting density attacks to low-weight knapsacks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 41–58. Springer, 2005.

[NS09] Phong Q Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[NS15] Phong Q Nguyen and Igor E Shparlinski. Counting co-cyclic lattices. *arXiv preprint arXiv:1505.06429*, 2015.

[NV10] Phong Q Nguyen and Brigitte Vallée. *The* LLL *algorithm*. Springer, 2010.

[Odl90] Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

[PG18] University of Bordeaux PARI Group. Pari-gp. `https://pari.math.u-bordeaux.fr/`, 2018. `https://pari.math.u-bordeaux.fr/gp.html`.

[Pis05] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[Pla05] Thomas Plantard. *Arithmétique modulaire pour la cryptographie.* PhD thesis, 2005. `https://documents.uow.edu.au/~thomaspl/pdf/Plantard05.pdf`.

[PMHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 685–716. Springer, 2019.

[Poh81] Michael Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM Sigsam Bulletin*, 15(1):37–44, 1981.

[PS09] Thomas Plantard and Willy Susilo. Broadcast attacks against lattice-based cryptosystems. In *Applied Cryptography and Network Security*, pages 456–472. Springer, 2009.

[PS10] Clément Pernet and William Stein. Fast computation of hermite normal forms of random integer matrices. *Journal of Number Theory*, 130(7):1675–1683, 2010.

[PS13] Colton Pauderis and Arne Storjohann. Computing the invariant structure of integer matrices: fast algorithms into practice. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, pages 307–314. ACM, 2013.

[PSDS18] Thomas Plantard, Arnaud Sipasseuth, Cédric Dumondelle, and Willy Susilo. DRS : Diagonal dominant reduction for lattice-based signature. PQC Standardization Conference, Round 1 submissions, 2018.

[PSSZ19] Thomas Plantard, Arnaud Sipasseuth, Willy Susilo, and Vincent Zucca. Tight bound on newhope failure probability. Cryptology ePrint Archive, Report 2019/1451, 2019. `https://eprint.iacr.org/2019/1451`.

[PSW08] Thomas Plantard, Willy Susilo, and Khin Than Win. A digital signature scheme based on CVP max. In *PKC 2008*, pages 288–307. Springer, 2008.

[PSZ12] Thomas Plantard, Willy Susilo, and Zhenfei Zhang. Lattice reduction for modular knapsack. In *International Conference on Selected Areas in Cryptography*, pages 275–286. Springer, 2012.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM (JACM)*, 51(6):899–942, 2004.

[RJL$^+$19] Avenzi Roberto, Bos Joppe, Ducas Léo, Klitz Eike, Lepoint Tancréde, Lyubaschevsky Vadim, Schanck John M., Schwabe Peter, Seiler Gregor, and Stehlé Damien. *CRYSTALS-Kyber (version 2.0) " Submission to round 2 of the NIST post-quantum project.*, 2.0 edition, 2019. `https://pq-crystals.org/kyber/resources.shtml`.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 1978.

[Saa17] Markku-Juhani O Saarinen. Hila5: On reliability, reconciliation, and error correction for ring-LWE encryption. In *International Conference on Selected Areas in Cryptography*, pages 192–212. Springer, 2017.

[Sch10] Claus Peter Schnorr. Average time fast SVP and CVP algorithms for low density lattices and the factorization of integers. Technical report, Tech. rep., Goethe Universität Frankfurt, 2010.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[Seb19] Jennifer Seberry. My fifty years of cryptography: the evolution of cryptographic standards. ACISP 2019, 2019. Invited Lecture, available at `http://www.uow.edu.au/~jennie/PublicationsAll.htm.old`.

[Sei18] Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-LWE lattice cryptography. Cryptology ePrint Archive, Report 2018/039, 2018. `https://eprint.iacr.org/2018/039`.

[SH95] Claus-Peter Schnorr and Horst Helmut Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–12. Springer, 1995.

[Sha82] Adi Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 145–152. IEEE, 1982.

[Sho97] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Slo83] N. J. A. Sloane. *Encrypting by Random Rotations*, pages 71–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[Söd11] Anders Södergren. On the poisson distribution of lengths of lattice vectors in a random lattice. *Mathematische Zeitschrift*, 269(3-4):945–954, 2011.

[SPS19a] Arnaud Sipasseuth, Thomas Plantard, and Willy Susilo. Enhancing Goldreich, Goldwasser and Halevi's scheme with intersecting lattices. *Journal of Mathematical Cryptology*, 13(3-4):169–196, 2019.

[SPS19b] Arnaud Sipasseuth, Thomas Plantard, and Willy Susilo. Improving the security of the DRS scheme with uniformly chosen random noise. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy*, pages 119–137, Cham, 2019. Springer International Publishing.

[SPS19c] Arnaud Sipasseuth, Thomas Plantard, and Willy Susilo. Using freivalds' algorithm to accelerate lattice-based signature verifications. In Swee-Huay Heng and Javier Lopez, editors, *Information Security Practice and Experience*, pages 401–412, Cham, 2019. Springer International Publishing.

[SPS20] Arnaud Sipasseuth, Thomas Plantard, and Willy Susilo. A noise study of the PSW signature family: Patching DRS with uniform distribution. *Information*, 11(3), 2020.

[SS00] Joan Serra-Sagristà. Enumeration of lattice points in l1 norm. *Information processing letters*, 76(1-2):39–44, 2000.

[ST04] Noah A Smith and Roy W Tromble. Sampling uniformly from the unit simplex. 2004.

[VAN94] MINGHUA QUAND SA VANSTONE. The knapsack problem in cryptography. *Finite Fields: Theory, Applications, and Algorithms: Theory, Applications, and Algorithms*, 168:291, 1994.

[Vau98] Serge Vaudenay. Cryptanalysis of the Chor-Rivest cryptosystem. In *Annual International Cryptology Conference*, pages 243–256. Springer, 1998.

[vdPS13] Joop van de Pol and Nigel P Smart. Estimating key sizes for high dimensional lattice-based systems. In *IMACC 2013*, pages 290–303. Springer, 2013.

[vEB81] P. van Emde Boas. *Another* NP*-complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice.* Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[WH10] Baocang Wang and Yupu Hu. Quadratic compact knapsack public-key cryptosystem. *Computers & mathematics with applications*, 59(1):194–206, 2010.

[Wil65] James Hardy Wilkinson. *The algebraic eigenvalue problem*, volume 662. 1965.

[WWH07] Baocang Wang, Qianhong Wu, and Yupu Hu. A knapsack-based probabilistic encryption scheme. *Information Sciences*, 177(19):3981–3994, 2007.

[YD17] Yang Yu and Léo Ducas. Second order statistical behavior of LLL and BKZ. In *International Conference on Selected Areas in Cryptography*, pages 3–22. Springer, 2017.

[YD18a] Yang Yu and Léo Ducas. Learning strikes again: The case of the DRS signature scheme. In *ASIACRYPT 2018*, pages 525–543. Springer, 2018.

[YD18b] Yang Yu and Léo Ducas. Learning strikes again: the case of the DRS signature scheme. Cryptology ePrint Archive, Report 2018/294, 2018.

[You09] Amr M Youssef. Cryptanalysis of a knapsack-based probabilistic encryption scheme. *Information Sciences*, 179(18):3116–3121, 2009.

[YY19] Masaya Yasuda and Junpei Yamaguchi. A new polynomial-time variant of LLL with deep insertions for decreasing the squared-sum of gram–schmidt lengths. *Designs, Codes and Cryptography*, pages 1–17, 2019.

[ZPL19] Yiming Zhu, Yanbin Pan, and Zhen Liu. When NTT meets karatsuba: Preprocess-then-NTT technique revisited. 2019. `https://eprint.iacr.org/2019/1079`.

[ZXZ$^+$18] Shuai Zhou, Haiyang Xue, Daode Zhang, Kunpeng Wang, Xianhui Lu, Bao Li, and Jingnan He. Preprocess-then-NTT technique and its applications to Kyber and new hope. In *International Conference on Information Security and Cryptology*, pages 117–137. Springer, 2018.

Appendix A

Code snippets

This appendix figures various code snippets, that people can use hopefully to test algorithms with MAGMA. While MAGMA is not free and open-source unlike SAGEMATH, OCTAVE, PARI-GP and other Computer Algebra System (CAS), it comes with a free online editor which is quite handy [CAG18] (note that at the time of testing a https connection is not available). We hope this can allow anybody to test computations with just an access to a web browser and an internet connection. This is especially useful for people without having access to admin rights in a public library computer or lightweight laptops with small processing power. To adapt to any screen size, you can change the size of the cells for input/output in the browser by dragging the lower-left corner.

Most of the code here is written considering a lower triangular **HNF** while MAGMA provides naturally upper-triangular **HNF**. This choice was made to match most of the academic literature we have came across. We believe that if a reader has the technical skills to use this code, they also have the technical skills to adapt it, therefore it should not be a problem. Furthermore, we have provided a code to switch to a lower-triangular **HNF** using MAGMA subroutines (see code in figure A.2).

Note that PARI-GP also has an online editor [PG18], but we have not tested much of it (yet). It seems MAGMA online does not require Javascript to be active and works well on mobile devices. PARI-GP needs Javascript. At the time of this thesis, MAGMA restricts “online” computations to 120 seconds, while PARI-GP does not provide a limit (but slow your browser down). That being said, if computations that lasts more than 2 minutes are needed, then using an online browser is probably not adapted: feel free to use any other CAS then.

The code presented in Figure A.8 was tested using the free version of MAGMA online at `http://magma.maths.usyd.edu.au/calc/`. At the time of the test, the

Figure A.1: Magma code for vector **HNF**-reduction with a Gauss-Jordan technique

```
// INPUT: "v" the vector to reduce, "H" the lower triangular HNF
// OUTPUT: Write in "w" the reduced form of "v" by "H"
ReduceByHNF:=procedure(~w,~v,~H)
   /* Initial values */
   NbCols:=Ncols(S);
   w:=v;
   i:=Ncols(S);

   /* Reduce */
   while i gt 0 do
      q:=Round(w[i]/H[i][i]);
      w:=w-(q*S[i]);
      i:=i-1;
   end while;
end procedure;
```

Figure A.2: Magma code for lower triangular **HNF**

```
// INPUT: A row matrix M
// OUTPUT: Replace M into its lower triangular HNF
LowerTriangleHNF:=procedure(~M)
   //Initial Swap
   NbCols:=Ncols(M);
   for i:=1 to Floor(NbCols/2) do
      SwapColumns(~M, i, NbCols-i+1);
   end for;
   //Compute HNF by trusting MAGMA
   M:=HermiteForm(M);
   //ReSwap Columns and rows
   NbCols:=Ncols(M);
   for i:=1 to Floor(NbCols/2) do
      SwapColumns(~M, i, NbCols-i+1);
   end for;
   NbRows:=Nrows(M);
   for i:=1 to Floor(NbRows/2) do
      SwapRows(~M, i, NbRows-i+1);
   end for;
end procedure;
```

MAGMA version was “V2.24-5”.

Figure A.3: Magma code for **PSW**-reduction

```
// INPUT: "v" a vector to reduce below "D" by "S" the reduction matrix
// OUTPUT: Write in "w" the reduced form of "v" by "S" undil "w < D"
ReduceDiag:=procedure(~w,~v,~S,D)
   /* Initial values */
   NbCols:=Ncols(S);
   i:=1;
   w:=v;

   /* Reduce */
   while Max([Abs(w[i]) : i in [1..NbCols]]) ge D do
      q:=Round(w[i]/S[i][i]);
      w:=w-(q*S[i]);
      if i eq NbCols then i:=0; end if;
      i:=i+1;
   end while;
end procedure;
```

Figure A.4: Magma code for the public key generation of **DRS**

```
/* Number of "obfuscation" rounds */
Rounds:=10;
/* Set P as the result of the R obfuscation of S with seed s */
PublicKeyDRS:=procedure(~P,~S,~s)
   /* Load constants */
   NbRows:=Nrows(S);
   Grp:=Sym(NbRows);
   P:=S;
   SetSeed(s);

   /* Apply R rounds of obfuscations */
   for i:=1 to Rounds do

      /* Random permutation */
      P:=PermutationMatrix(Integers(), Random(Grp))*P;

      /* Multiplication by unimodular matrix */
      for j:=1 to NbRows-1 by 2 do
         sgn:=2*Random(1)-1;
         P[j]:= P[j] + (sgn * P[j+1]);
         P[j+1]:= P[j+1] + (sgn * P[j]);
      end for;

   end for;
   P:=PermutationMatrix(Integers(), Random(Grp))*P;
end procedure;
```

Figure A.5: Magma code for the signature of **DRS**

```
/* given a DRS secret key/seed S/s, find kP=v-w with w < D */
ReduceDRS:=procedure(~k,~w,~v,~S,~s,D)
  /* Initialize constants */
  NbCols:=Ncols(S);
  NbRows:=Nrows(S);
  Grp:=Sym(NbRows);
  P:=S;
  i:=1;
  w:=v;
  k:=Vector([0 : i in [1..NbCols] ]);

  /* Reduce the vector to kS=v-w */
  while Max([Abs(w[i]) : i in [1..NbCols]]) ge D do
    /* Depending on the noise used, switch between division by D or the
       diagonal coefficient of the whole matrix */
    //q:=Round(w[i]/S[i][i]);
    q:=Round(w[i]/D);
    k[i]:=k[i]+q;
    w:=w-(q*S[i]);
    if i eq NbCols then i:=0; end if;
    i:=i+1;
  end while;

  /* Transform kS=v-w to kP=v-w */
  SetSeed(s);
  for i:=1 to Rounds do
    k:=k*Transpose(PermutationMatrix(Integers(), Random(Grp)));
    for j:=1 to NbRows-1 by 2 do
      sgn:=2*Random(1)-1;
      k[j+1]:= k[j+1] - (sgn * k[j]);
      k[j]:= k[j] - (sgn * k[j+1]);
    end for;
  end for;
  k:=k*Transpose(PermutationMatrix(Integers(), Random(Grp)));
end procedure;
```

Figure A.6: Magma code for computing the max norm of a matrix

```
/* put in res the max norm of mat with ln lines and col coloumns*/
MaxMatNorm:=procedure(~res,~mat,~ln,~col)
   res:=0;
   for i:=1 to col do
      tmp:=0;
      for j:=1 to ln do
         tmp:=tmp+Abs(mat[j][i]);
      end for;
      res:=Maximum(res,tmp);
   end for;
end procedure;
```

Figure A.7: Magma code for the verification in **DRS**

```
/* Put in Bool whether kP=v-w and w < D */
VerifyDRS:=procedure(~Bool,~k,~w,~v,~P,D)
  /* Initialize constants */
  NbCols:=Ncols(P);
  NbRows:=Nrows(P);
  /*Use B:=2 for speed, B:=10 is for visual representation*/
  B:=10;
  Zero:=Vector([0 : i in [1..NbCols]]);
  End:=true;

  /* Initialize loop parameters */
  q:=k;
  t:=v-w;
  modulo:=0;
  MaxMatNorm(~modulo,~P,~NbRows,~NbCols);
  modulo:=B^Floor(Log(B,modulo));

  /* Checks the max norm */
  Bool:=Max([Abs(w[i]) : i in [1..NbCols]]) lt D;
  if (not Bool) then End:=false;Bool:=false; end if;

  while Bool do
    /*Check r <- load part of q */
    r:=Vector([Round(q[i]/modulo) : i in [1..NbCols]]);
    r:=Vector([q[i] - (r[i]*modulo) : i in [1..NbCols]]);
    t:=t-(r*P);

    /* Check equality for that block */
    t2:=Vector([t[i] mod modulo : i in [1..NbCols]]);
    if (t2 ne Zero) then
      End:=false;break;
    end if;

    /* Eliminate block and update values */
    t:=Vector([ ExactQuotient(t[i], modulo) : i in [1..NbCols]]);
    q:=Vector([ ExactQuotient(q[i]-r[i], modulo) : i in [1..NbCols]]);
    q_not_zero:=(q ne Zero);
    t_not_zero:=(t ne Zero);

    /* Test if one component is prematurely zero */
    if ((not q_not_zero) xor (not t_not_zero)) then
      End:=false;break;
    end if;

    /* Test if all components are zero: if yes we finished */
    Bool:= (q_not_zero) and (t_not_zero);
  end while;
  Bool:=End;
end procedure;
```

Figure A.8: Magma code for testing both **DRS** and **PSW** conditions

```
/* Set Randomness, Diagonal Coefficient and Dimension */
Seed:=1515430315;
SetSeed(Seed);
N:=51;
D:=N;

/* Initialize to Real values for computation of the Spectral Radius */
M:=ZeroMatrix(GetDefaultRealField(),N,N);

/* Randomly put noise values */
for i:=1 to N do
   for j:=1 to N do
      /* High probability of respecting PSW-bound but not DRS-bound */
      M[i,j]:=Random(0,3)*((Random(0,1)*2)-1);
      /* High probability of respecting DRS-bound but not PSW-bound */
      // M[i,j]:=Random(0,Ceiling(D/N));
   end for;
end for;

/* Compute Spectral Radius to check validity of PSW Conjecture */
SR:=SpectralRadius(M)*(D^-1);

/* Check the norm l1 for each vector of the noise */
MinS:=2*D;
MaxS:=0;
AvgS:=0;

for i:=1 to N do
   S:=0;
   for j:=1 to N do
      S:=S+Abs(M[i,j]);
   end for;
   if S gt MaxS then MaxS:=S; end if;
   if MinS gt S then MinS:=S; end if;
   AvgS:=AvgS+S;
end for;

AvgS:=AvgS/N;

/* Print Results */
print "Random Seed is " cat IntegerToString(Seed);
print "Diagonal Value D is " cat IntegerToString(D);
print "Dimension N is " cat IntegerToString(N);
print "";
print "Spectral Radius";SR;
print "Minimum/Maximum l1 norm of noise vectors";
print Floor(MinS),Floor(MaxS);
print "Average l1 norm of noise vectors";
print Floor(AvgS);
```

Figure A.9: Magma code for playing around **HNF** intersections: general case

```
// General MAGMA method for intersections
M1:=Matrix([
[32,13,15],
[14,17,13],
[87,21,11]
]);

//HermiteForm(M1);

M2:=Matrix([
[2,13,1],
[12,98,11],
[7,21,21]
]);

//HermiteForm(M2);

L1:=Lattice(M1);
L2:=Lattice(M2);

LInter:=L1 meet L2;
MInter:=Basis(LInter);

print "Expected Result using MAGMA command \"meet\"";
print HermiteForm(Matrix(MInter));

// Other general method without dual computation, using HNF algorithms
P1:=HorizontalJoin(M1,M1);
//print "P1",P1;
P2:=HorizontalJoin(M2,ZeroMatrix(Integers(),Nrows(M2),Ncols(M1)));
//print "P2",P2;

PJoin:=VerticalJoin(P1,P2);
//print "PJoin pre HNF",PJoin;

PJoin:=HermiteForm(PJoin);
//print "PJoin post HNF",PJoin;

PJoin:=ExtractBlock(PJoin,Nrows(M1)+1,Ncols(M1)+1,Nrows(M2),Ncols(M2));
print "Result M1 inter M2 using Hermite Normal Forms",PJoin;
```

Figure A.10: Magma code for playing around **HNF** intersections: q-ary case

```
// Define 4 matrices with a single column and the same determinant
M1:=Matrix([
[1,0,0,0,0,0],
[0,1,0,0,0,2],
[0,0,1,0,0,0],
[0,0,0,1,0,0],
[0,0,0,0,1,0],
[0,0,0,0,0,5]]);
M2:=Matrix([
[1,0,0,0,0,0],
[0,1,0,0,3,0],
[0,0,1,0,0,0],
[0,0,0,1,0,0],
[0,0,0,0,5,0],
[0,0,0,0,0,1]]);
M3:=Matrix([
[1,0,0,0,0,0],
[0,1,0,3,0,0],
[0,0,1,0,0,0],
[0,0,0,5,0,0],
[0,0,0,0,1,0],
[0,0,0,0,0,1]]);
M4:=Matrix([
[1,0,0,0,0,0],
[0,1,2,0,0,0],
[0,0,5,0,0,0],
[0,0,0,1,0,0],
[0,0,0,0,1,0],
[0,0,0,0,0,1]]);

// Generate all related lattices
L1:=Lattice(M1);
L2:=Lattice(M2);
L3:=Lattice(M3);
L4:=Lattice(M4);

// Intersect them all
LInter:=L1 meet L2;
LInter:=LInter meet L3;
LInter:=LInter meet L4;

// Observe the resulting q-ary lattice
HermiteForm(Matrix(Basis(LInter)));
```

Figure A.11: Magma code for playing around **DRS**

```
// Declare the secret key with D=10
S:=Matrix([
[10,0,2,-3,0,1],
[-1,10,2,3,0,2],
[1,0,10,3,0,-1],
[0,-4,2,10,0,3],
[-1,0,2,3,10,-2],
[3,3,0,-1,0,10]
]);

// Create the public key
P:=S;
s:=3;
PublicKeyDRS(~P,~S,~s);

// Create a large vector and sign it
v:=Vector([Random(1000) : i in [1..6]]);
w:=v;
k:=v;
ReduceDRS(~k,~w,~v,~S,~s,10);

// Print the resulting computations
print "Secret Key:", S;
print "Public Key:", P;
print "v to reduce:", v;
print "signature k,w:";
print k,w;
print "v-w",v-w;
print "k*P",k*P;

// Verify the result
Bool:=true;
VerifyDRS(~Bool,~k,~w,~v,~P,10);
print "Check:", Bool;
k*P+w-v;
```

Figure A.12: MAGMA code for playing around Freivalds' algorithm

```
// Apply one round of the Freivalds test
FreivaldsOneRound:=procedure(~res,~A,~B,~C)
  v:=Vector([Random(0,1) : i in [1..Ncols(A)]]);
  vc:=v*C;
  v:=v*A;
  v:=v*B;
  res:=(v eq vc);
end procedure;

Rounds:=3;
// Apply multiple rounds of the Freivalds test
Freivalds:=procedure(~res,~A,~B,~C)
  tmp:=true;
  for i:=1 to Rounds do
    FreivaldsOneRound(~res,~A,~B,~C);
    tmp:=(tmp and res);
  end for;
  res:=tmp;
end procedure;

//Fix dimension, bound of values
N:=5;
BoundVal:=2;

//Generate samples, with one correct answer and one not
A:=ZeroMatrix(Integers(),N,N);
B:=ZeroMatrix(Integers(),N,N);
FakeAB:=ZeroMatrix(Integers(),N,N);

for i:=1 to 5 do
  A[i]:=Vector([Random(-BoundVal,BoundVal) : i in [1..N]]);
  B[i]:=Vector([Random(-BoundVal,BoundVal) : i in [1..N]]);
  FakeAB[i]:=Vector([Random(-BoundVal*N,BoundVal*N) : i in [1..N]]);
end for;

AB:=A*B;

print "A",A;
print "B",B;

print "AB",AB;
print "FakeAB",FakeAB;

res:=0;

// Test both the correct answer and the fake
Freivalds(~res,~A,~B,~AB);
print "Test with AB: ",res;

Freivalds(~res,~A,~B,~FakeAB);
print "Test with FakeAB: ",res;
```

Figure A.13: MAGMA code for playing around our modified Freivalds for DRS

```
S:=Matrix([
[10,0,2,-3,0,1],
[-1,10,2,3,0,2],
[1,0,10,3,0,-1],
[0,-4,2,10,0,3],
[-1,0,2,3,10,-2],
[3,3,0,-1,0,10]
]);

// Create the public key
P:=S;s:=3;
PublicKeyDRS(~P,~S,~s);
print "Public Key",P;

/* Pick 2 primes (could be more) */
PrimeSize:=12;
p1:=RandomPrime(PrimeSize); p2:=RandomPrime(PrimeSize);
print "pick 2 primes:",p1,p2;

/* Create one vector per prime */
x1:=Matrix([[Random(0,p1-1) : i in [1..6]]]);
x2:=Matrix([[Random(0,p2-1) : i in [1..6]]]);

/* Precompute a multiplier */
V:=HorizontalJoin(Transpose(x1),Transpose(x2));
print "precomputed multiplier x for (v-w):",V;

/* Precompute product with public key */
T:=P*V;
for i:=1 to Nrows(T) do
   T[i,1]:= T[i,1] mod p1; T[i,2]:= T[i,2] mod p2;
end for;
print "precomputed multiplier X for k:",T;

// Create a large vector and sign it
v:=Vector([Random(1000) : i in [1..6]]); w:=v; k:=v;
ReduceDRS(~k,~w,~v,~S,~s,10);

/* From here apply the new verification process */

kt:=k*T; kt[1]:= kt[1] mod p1; kt[2]:= kt[2] mod p2;
print "result kX:",kt;

vt:=(v-w)*V; vt[1]:= vt[1] mod p1; vt[2]:= vt[2] mod p2;
print "result vx:",vt;

print "Success (kx = vx): ",(vt eq kt);
```