Gaussian Sampling in Lattice-Based Cryptography

Thomas Prest

Ph.D. Thesis

Gaussian Sampling in Lattice-Based Cryptography

Thèse de doctorat présentée et soutenue publiquement le 8 décembre 2015 par

Thomas Prest

pour obtenir le grade de

Docteur de l'École Normale Supérieure

Comité de thèse

Directeurs de thèse :	David Pointcheval	École Normale Supérieure
	Vadim Lyubashevsky	IBM Zürich
Encadrant Industriel :	Sylvain Lachartre	Thales Communications & Security
Rapporteurs :	Eike Kiltz	Ruhr-Universität Bochum
	Damien Stehlé	École Normale Supérieure de Lyon
Président du jury :	Jean-Sébastien Coron	Université du Luxembourg

THALES
Thales Communications & Security
Laboratoire Chiffre

ENS
École Normale Supérieure
Équipe Crypto

Remerciements

"A Lannister pays his debts."/"Un Lannister paye ses dettes."

Tyrion Lannister — George R. R. Martin. *A Song of Ice and Fire*

I would first like to thank my advisor, Vadim Lyubashevsky. Adapting to his way of research was tough, but fun and stimulating. I learned a lot from him, in particular career-defining skills that I couldn't have learned in books.

Je remercie David Pointcheval, directeur de l'équipe Crypto de l'École Normale Supérieure, et Eric Garrido, directeur du laboratoire Chiffre de Thales, de m'avoir accueilli dans leurs laboratoires respectifs. Grâce à leur bienveillance, j'ai toujours bénéficié de conditions idéales pour effectuer ma thèse, et ce au sein de mes deux équipes.

I would like to thank Eike Kiltz and Damien Stehlé for reviewing my thesis. I know it was a lot of work, and I am fortunate that they agreed to do it. J'aimerais également remercier Jean-Sébastien Coron d'avoir accepté de faire partie de mon jury.

J'aimerais remercier mon encadrant de Thales, Sylvain Lachartre. Grâce à lui et à Olivier Orcière, j'ai pu réaliser un stage en entreprise malgré mon profil très académique à l'époque. Puis lors de ma thèse, ses nombreux conseils m'ont été très utiles. Un grand merci également aux autres membres du laboratoire Chiffre: les permanents Alexandre, David, Didier, Émeline, Éric, Olivier Bernard, Philippe, Renaud, Sonia et les (anciens) doctorants Aurore, Christopher, Julia et Thomas. J'apprécie beaucoup l'ambiance chaleureuse de l'équipe et c'est un plaisir de la rejoindre en tant qu'ingénieur cryptologue dès janvier prochain.

J'ai également eu la chance d'effectuer ma thèse à l'ENS et d'être dans un milieu propice à la recherche, parmi des thésards et chercheurs brillants: Adrian, Alain, Anca, Angelo, Antonia, Aurore, Balthazar, Céline, Dahmun, Damien, Édouard, Fabrice, Florian, Geoffroy, Guiseppe, Hoeteck, Houda, Itai, Jérémy, Léo, Liz, Louiza, Mario Cornejo, Mario Strefler, Michel, Michele, Nuttapong, Pierre-Alain, Phong, Pierrick, Rafael, Razvan, Romain, Simon, Sonia, Sorina, Sylvain, Tancrède, Thierry, Thomas et Yuanmi. J'aimerais tout particulièrement remercier Aurore pour ses conseils avisés, et j'ai beaucoup apprécié discuter de réseaux avec Tancrède, Rafael, Pierrick (qui, de manière assez paradoxale, contribue beaucoup à l'âme du labo) et mon inénarrable multi-coauteur Léo. J'ai également pu parler d'algèbre de Boole avec Adrian le spécialiste des petits fours, de PRF dans le modèle standard avec mon voisin de bureau Alain, et de la vie avec Mario.

Les évènements auxquels j'ai assisté – notamment les journées C2 que je recommande à tout doctorant en cryptographie – m'ont permis de rencontrer des thésards et jeunes chercheurs sympathiques et intelligents. J'ai pris plaisir à discuter ou prendre (plus d')un verre avec Jean-Christophe Deneuville, Malika Izabachene, Adeline Langlois, Cécile Pierrot, Frédéric de Portzamparc, Ange Martinelli, Vincent Grosso, Romain Poussier, Guillaume Bonnoron, Pierre Karpman, Fabrice Mouhartem, Marie Paindavoine, Julien Devigne, Raphaël Bost et bien d'autres, et si l'occasion se présente, je serai également ravi de travailler avec eux. Merci également à Carlos Aguilar pour m'avoir invité quelques jours à Toulouse, et pour sa librairie NFLlib qui m'a permis d'obtenir des résultats inespérés.

J'aimerais tout particulièrement remercier Damien Stehlé pour m'avoir reçu au sein de son équipe à Lyon, pour ses remarques pertinentes qui ont très grandement amélioré la qualité de certains passages de ma thèse, et pour avoir toujours été disponible malgré son emploi du temps chargé. De manière plus générale, sa rigueur mathématique est bénéfique pour la cryptographie, et l'exposé "Santa Claus doesn't exist" restera mon souvenir le plus mémorable de ces trois années de thèse.

I would like to thank the rest of the team AriC of ENS Lyon: Benoît, Fabien, Guillaume, Gilles, Jinming, Sanjay, Sébastien, Shi, Somindu, Valentina and Weiqiang, as well as Jung-Hee and Yong-Soo who were visiting at the same time as me. My stay at Lyon was short but insightful, and it was a pleasure to exchange ideas with them.

J'en profite pour remercier pour leur aide précieuse Joëlle Isnard, Michelle Angely, Lise-Marie Bivard, Valérie Mongiat de l'ENS, Lydie Pezant de Thales, Chiraz Benamor de l'ENS Lyon et tout particulièrement Nathalie Gaudechoux de l'INRIA. Grâce à leur excellent travail, j'ai pu naviguer sans encombre entre les couches du millefeuille administratif.

J'aimerais également remercier mon directeur de stage de Master I en 2010, Paul Zimmermann, ainsi que toute l'équipe CARAMEL de Nancy de l'époque: Pierrick Gaudry, Emmanuel Thomé, Jérémie Detrey, Nicolas Estibals, Gaëtan Bisson, et j'en oublie malheureusement sûrement. Ce stage fut une très bonne expérience et m'a décidé à suivre une carrière en cryptographie.

Merci à Pierrick, Léo et ma soeur Céline d'avoir relu une version préliminaire de ma thèse.

Pendant mon Master II à Bordeaux, j'ai eu le plaisir de rencontrer des camarades de promos qui sont devenus des amis, et avec qui j'ai pu découvrir ensemble les joies – métros bondés, serveurs sarcastiques, restaurants de bobos, hipsters sur vélo à pignon fixe – de Paris: Damien, Guillaume, notre maman Marie-Luce, Olivier, notre doyen Pierre-Arnaud, Romain, ainsi que leurs pièces rapportées Mandine, Nicolas, Charlotte, Floriane et Malorie. J'espère ne pas avoir été trop insupportable pendant ces trois ans (je ne garantis pas que ça s'améliorera).

Enfin, j'aimerais remercier ma famille nucléaire+ϵ: mes parents, ma soeur Céline et son mari Daniel. Sans eux, jamais je n'aurais pu arriver jusque là.

Contents

List of Algorithms

Notations

This thesis contains a fairly large number of notations. To make its comprehension easier to the reader, the following tables summarize all the notations we use.

Acronyms

Notation	Brief definition	See also
BLISS	Bimodal Lattice Signature Scheme.	[DDLL13]
C&C	Command and control.	Page 159
CCA	Chosen-ciphertext attack.	–
CPA	Chosen-plaintext attack.	–
CVP	Closest vector problem.	Page 20
ECC	Error-correcting code.	Page 148
FDH	Full-domain hash (signature).	Page 37
FRG	Full-rank Gram (matrix).	Page 121
FFT	Fast Fourier transform.	[Nus12]
GSD	Gram-Schmidt decomposition.	Page 66
GSO	Gram-Schmidt orthogonalization.	Page 23
GSR	Gram-Schmidt reduction	Page 65
GPV	Gentry-Peikert-Vaikuntanathan. Denotes the signature and IBE framework of [GPV08].	Page 37
IBE	Identity-based encryption (scheme).	Page 39
IBS	Identity-based signature (scheme).	Page 156
KL	Kullback-Leibler (divergence).	Page 45
LLL	Lenstra-Lenstra-Lovász (algorithm).	Page 29
LWE	Learning wirh errors (problem).	Page 16
NIKE	Non-interactive key exchange.	Page 156
NTRU	"N-th degree TRUncated polynomial ring" OR "Number Theory Research Unit" OR "Number Theorists R Us". Usually denotes a cryptosystem [HPS98] or, as in this thesis, a class of lattices.	Page 100
PKI	Public key infrastructure.	Page 155
SIS	Short integer solution (problem).	Page 16
SVP	Shortest vector problem.	Page 20
UAKE	Unidirectional authenticated key exchange	Page 157

Generic Notations

Notation	Brief definition	See also
$\triangleq$	$a \triangleq b$ means "a is defined as b".	Page 19
$\mathbb{C}/\mathbb{R}/\mathbb{Q}/\mathbb{Z}/\mathbb{N}$	Complex/real/rational/integers/non-negative integers.	Page 19
λ	A security parameter (typically 80, 128 or 192).	–
$\lfloor a \rceil$	Rounds a real number to a closest integer.†	Page 19
$a^\star$	The conjugate of $a \in \mathbb{R}[x]/(h(x))$.†	Page 117
log	The natural logarithm.	Page 19
$O, \tilde{O}, o, \omega, \Omega, \sim$	Asymptotic comparison notations.	Page 19
$V^\perp$	The orthogonal complement of $V^\perp$.	Page 21
$\mathbf{Proj}(\mathbf{b}, V)$	Projects a vector $\mathbf{b}$ onto a vector space V.	Page 22
H	A vector space over some field $\mathbb{K}$.	Page 19
$\langle \cdot, \cdot \rangle$	A dot product $H \times H \to \mathbb{R}$.	Page 19
$\| \cdot \|$	A norm $H \to \mathbb{R}^+$.	Page 19
Λ	A lattice.	Page 20
$\lambda_i(\Lambda)$	The i-th successive minima of Λ.	Page 20
$\det(\Lambda)$	The determinant of a lattice Λ.	Page 20
$\Lambda_q^\perp$	The orthogonal lattice of Λ mod q.	Page 21
$\mathrm{Span}_X(\mathbf{B})$	The set of finite combinations $\sum_{(x_i, \mathbf{b}_i) \in X \times \mathbf{B}} x_i \cdot \mathbf{b}_i$.	Page 19
$\mathcal{L}(\mathbf{B})$	The lattice spanned by the vectors of $\mathbf{B}$.	Page 20
$\mathbf{B}_k$	The k first vectors of $\mathbf{B}$.	Page 23
$\mathbf{B}^\star$	The conjugate transpose matrix of $\mathbf{B}$.	Page 25
$\mathbf{B}^+$	The Moore-Penrose pseudoinverse of $\mathbf{B}$.	Page 22
$\tilde{\mathbf{B}}$	The Gram-Schmidt orthogonalization of $\mathbf{B}$.	Page 23
$\lvert\tilde{\mathbf{B}}\rvert$	The Gram-Schmidt norm of $\mathbf{B}$.	Page 25
$s_1(\mathbf{B})$	The largest singular value of $\mathbf{B}$.	Page 25
$\mathcal{P}(\mathbf{B})$	The fundamental parallelepiped generated by $\mathbf{B}$.	Page 25
$\mathbf{L}$	A matrix such that $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$.$^\heartsuit$	Page 65
Δ	The statistical distance.	Page 31
$\rho_{\sigma, \mathbf{c}}$	The Gaussian function of standard deviation σ, centered on the point $\mathbf{c}$.	Page 31
$D_{\Lambda, \sigma, \mathbf{c}}$	A discrete Gaussian distribution over Λ, parameterized by σ and $\mathbf{c}$.	Page 31
D_1	The continuous spherical Gaussian centered on $\mathbf{0}$ and of standard deviation 1.	Page 31
$\lfloor c \rceil_\sigma$	The randomized rounding $D_{\mathbb{Z}, \sigma, c}$.†	Page 32
$\eta'_\epsilon(\Lambda)$	The smoothing parameter of Λ.	Page 32
$x \xleftarrow{\$} E$	x sampled from the uniform distribution over E.	–
ζ_m	A complex primitive m-th root of unity.	Page 66
Ω_m	The set of complex primitive m-th roots of unity.	Page 66
ϕ_m	The m-th cyclotomic polynomial.	Page 66
$\mathbb{Z}_m^\times$	The group of invertible elements of $\mathbb{Z}_m \triangleq \mathbb{Z}/m\mathbb{Z}$.	Page 66
$\varphi(m)$	Euler's totient function on m: $\lvert\mathbb{Z}_m^\times\rvert$.	Page 66
$\mathcal{A}_\phi(f)$	The anticirculant matrix associated to ϕ and f.	Page 67

Notations Specific to Chapter 4

Notation	Brief definition	See also
r	An isometry	Page 64
$\mathbf{v}_k$	$\mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, r(\mathrm{Span}(\mathbf{B}_{k-1})))$.$^\heartsuit$	Page 68
C_k, D_k	$C_k = \langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k \rangle$ and $D_k = \|\tilde{\mathbf{b}}_k\|^2$.$^\heartsuit$	Page 70
$\lambda_{i,j}, \mathbf{db}_j, \mathbf{dv}_j, c_j, d_j$	Integral variants of $\det(\mathbf{B}_j), \mathbf{b}_j, \mathbf{v}_j, C_j, D_j$.$^\heartsuit$	Page 74

$^\heartsuit$We assume that $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ is clear from context.

Notations Specific to Chapters 5, 6 and 8

Notation	Brief definition	See also
$\mathbb{K}$	A number field.	Page 88
$\mathbb{K}^+$	The maximal real subfield of $\mathbb{K}$.	Page 88
$\mathcal{R}$	The ring of integers of $\mathbb{K}$, in practice $\mathbb{Z}[x]/(x^N + 1)$.$^\infty$	Page 88
$\langle \cdot, \cdot \rangle_{\mathbb{K}}$	A dot product $H \times H \to \mathbb{K}$.	Page 88
$\| \cdot \|_{\mathbb{K}}$	A norm $H \to \mathbb{K}^+$.	Page 88
$\|\tilde{\mathbf{B}}\|_{\mathbb{K}}$	The generalized Gram-Schmidt norm of $\mathbf{B}$.	Page 89
$\mathcal{Z}$	The ring $\mathbb{Z}[x]/(x^N + 1)$.$^\infty$	Page 100
$\mathcal{K}$	The field $\mathbb{Q}[x]/(x^N + 1)$.$^\infty$	Page 100

$^\infty$We assume that N is implicit from context.

Notations Specific to Chapter 7

Notation	Brief definition	See also
$\mathcal{R}_d$	The convolution ring $\mathbb{R}[x]/(x^d - 1)$.	Page 117
$c(a)$	The coefficients' vector of a.†	Page 118
$\mathcal{C}(a)$	The circulant matrix of a.†	Page 118
$\mathrm{gpd}(d)$	The greatest proper divisor of d.	Page 118
$\mathsf{V}_{d\backslash d'}(a)$	The vectorization operator. "Breaks" $a \in \mathcal{R}_d$ into a vector in $\mathcal{R}_{d'}^{d/d'}$.†	Page 118
$\mathsf{M}_{d\backslash d'}(a)$	The matrixification operator. "Breaks" $a \in \mathcal{R}_d$ into a matrix in $\mathcal{R}_{d'}^{(d/d')\times(d/d')}$.†	Page 119
$\mathcal{Z}_d$	The ring $\mathbb{Z}[x]/(x^d - 1)$.	Page 127
ψ_d	The polynomial $\prod_{\zeta^d=1, \zeta \notin \Omega_d}(x - \zeta)$.	Page 129
$\mathcal{F}_d$	The cyclotomic ring $\mathbb{R}[x]/(\phi_d(x))$.	Page 129
ι_d	An inner-product preserving embedding of $\mathcal{F}_d$ in $\mathcal{R}_d$.	Page 129

†This operation extends to vectors and/or matrices.

Chapter 1

Introduction

"No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems unlikely that anyone will do so for many years."

— G. H. Hardy. *A Mathematician's Apology, 1940*

Introduction to Cryptology

Cryptology encompasses two different but closely intricate fields of studies. The first one is cryptography, which aims to hide informations and protect communications. The second one is cryptanalysis, and its goal is to recover or compromise information protected by cryptography. This makes cryptology a dual science in essence which, like the mythological snake Ouroboros, constantly feeds on itself.

Although the need to secure communications started more than two millenia ago, prompted by military use, cryptology remained little studied and more an art that a science until recently. In *La Cryptographie Militaire* [Ker83], Augustus Kerckhoffs edicted six principles that layed some foundations for modern cryptography. The most important one stated that a cryptographic scheme should remain secure even if its entire design is known, as long as the key is kept secret.[1]

Cryptology really began to develop at the beginning of the XXth century, fueled by the appearance of rudimentary machines and the increasing need for securing communications. But it is arguably the Enigma machine that sparked the entry of cryptology in the modern era. Created in the early 1920's by a German engineer, it was extensively used by the German army to encrypt their communications during World War II. The repeated efforts of Polish and British mathematicians to break it, as well as the numerous modifications subsequently done on the German side, marked the first effort on an industrial scale to provide strong cryptography and efficient cryptanalysis, and the entry of cryptolophy in the modern era.

1.1 Public Key Cryptography

Perhaps the most important development in the practice of cryptography is the paper of Whitfield Diffie and Martin E. Hellman, *New Directions in Cryptography* [DH76], published

[1] Despite the widespread consensus around this principle and standard bodies recommending it, many protocols continue to partially rely on security through obscurity, with often disastrous outcomes. Two examples are the CSS system for protecting DVDs and A5/1, A5/2 protocols for GSM communications.

in 1976. Until then, essentially all encryption schemes had been using symmetric key algorithms, which required the prior agreement of both parties on a *common secret key*. In some cases, this led to cryptographic keys being sent via extremely impractical means such as trusted couriers and diplomatic bags and made secure communication out of reach for common people. With Diffie and Hellman's protocol, two parties can securely agree on a common private key even if they communicate on a completely insecure channel. The following year, Ronald L. Rivest, Adi Shamir and Leonard M. Adleman published *A Method for Obtaining Digital Signature and Public-Key Cryptosystems* [RSA78], where they described how to sign electronic documents and send encrypted messages without the need to share any common key. More precisely, by sharing their *public* keys, Bob can send an encrypted message that can be read by Alice only, and conversely Alice has the guarantee that no other than Bob wrote the message she received, all of this on an unencrypted channel. Along with Diffie and Hellman's paper, the work of Rivest, Shamir and Adleman spawned a new branch of cryptology that soon became known as public key cryptography and now has countless applications.

Public key cryptography relies heavily on one-way functions, that are functions easy to evaluate on any input but hard to invert given an output. Most of the one-way functions used in this context rely on the hardness of number theory problems and as a result, cryptography and number theory have become closely connected. A quote by Hendrik Lenstra in 2002 summarizes well the current situation:

> Nowadays, when a number theorist applies for a grant, he says that number theory is used in cryptography, and so doing number theory is good for national security. Back then, since if was before the discovery of America, they said number theory is used in music. But I won't comment on the progress of civilization since then.

which stands in stark contrast with the opening quote of G. H. Hardy. Over the years, number theory proved to be a useful weapon to create one-way functions and trapdoor functions, which similarly to one-way functions are easy to compute and hard to invert *except when provided some special information*. Notably, hardness of problems in modular arithmetic [Rab79, RSA78], cyclic groups [ElG84] and elliptic curves [Mil86, Kob89] have provided very efficient constructions and continue to actively fuel cryptographic research to this day.

1.2 Lattice-Based Cryptography

Lattices are repeating arrangements of points in a grid pattern. Figure 1.1 represents a two-dimensional lattice, but they can be defined in any dimension strictly bigger than 0. Lattices are infinite sets but for any lattice Λ, there exist *finite* sets of the form $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ such that Λ can be represented by the set of integer combinations $z_1\mathbf{b}_1 + ... + z_n\mathbf{b}_n$.

A breakthrough work of Ajtai [Ajt96] in 1996 introduced the Short Integer Solution problem, or SIS problem. He demonstrated how to construct instances of this problem and unveiled connections between worst-case "natural" problems over lattices and the average-case SIS problem. Around the same time, Hoffstein, Pipher and Silverman [HPS98] presented NTRU, an extremely fast public key encryption scheme based on polynomial rings but against which all practical attacks are based on lattices. Both works started the construction of cryptographic primitives using lattices, or what is now called lattice-based

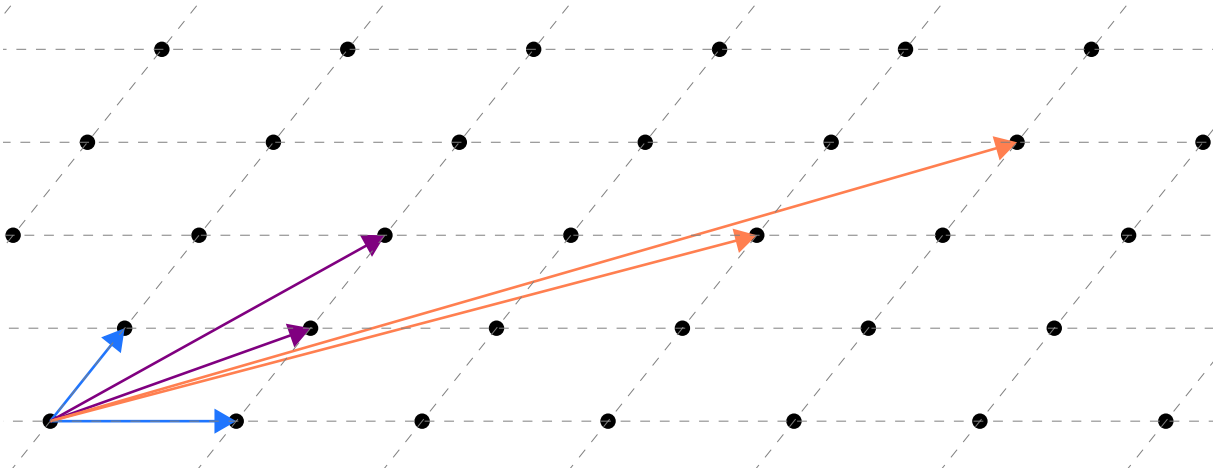


Figure 1.1: A lattice and three bases.

cryptography. In 2005, Regev [Reg05] introduced the Learning With Errors problem, or LWE problem, which was also shown to be as hard as standard lattice problems. Both Ajtai's SIS and Regev's LWE problems proved to be extremely versatile building blocks for a wide array of cryptographic schemes, including fully homomorphic encryption, a construction which allows to perform an arbitrary number of operations on encrypted data without having to know anything about the data.

Additionally, lattice-based schemes can rely on ring lattices. These have the same form $\Lambda = \{z_1\mathbf{b}_1 + ... + z_n\mathbf{b}_n\}$ as standard lattices, but instead of being in $\mathbb{Z}$, the z_i's and the coefficients of the vectors $\mathbf{b}_i$ are in some specific ring $\mathcal{R}$. Concurrently introduced by Lyubashevsky and Micciancio [LM06] and Peikert and Rosen [PR06], a series of works [LM06, PR06, SSTX09, LPR10, LPR13b] showed that generalized versions of SIS and LWE, called Ring SIS and Ring LWE (or simply RSIS and RLWE), admitted worst-case reductions from standard lattice problems over *ideal* lattices (which are ideals of the ring of integers of a number field). This allows efficient instantiations of lattice-based schemes.

1.2.1 Gaussian Sampling

A powerful tool in lattice-based is Gaussian sampling. Introduced in cryptography by Gentry, Peikert and Vaikuntanathan [GPV08] in 2008, it consists, given an arbitrary point $\mathbf{c}$ and a short basis $\mathbf{B}$ of a lattice Λ, to sample a point according to a Gaussian distribution discretized on the lattice points and centered on $\mathbf{c}$. A key point of Gaussian sampling is that it doesn't leak any information about the lattice, effectively behaving like an oracle – although a weaker one than an oracle which returns the closest point to $\mathbf{c}$. Under the hypothesis that SIS and LWE are hard, [GPV08] showed how to use Gaussian sampling to create lattice-based signature schemes, as well as lattice-based identity-based encryption, a paradigm of public key encryption where a user's public key is its own identity, resolving many key-distribution problems.

Gaussian sampling has since allowed to realize advanced functionalities, such as hierarchical identity-based encryption schemes [CHKP10, ABB10b], standard-model signatures [ABB10b, Boy10b], attribute-based encryption [BGG+14], and many other constructions. Interestingly, by itself, the very problem of doing Gaussian sampling has also showed recent connections with standard lattice problems [ADRS15, ADS15, Ste15].

1.3 This Thesis

The versatility of Gaussian sampling as a black box tool is recognized and accordingly used to create many constructions. However, except for [Kle00, GPV08, Pei10] and in some sense [MP12], no work we are aware of proposes generic Gaussian samplers and few works propose improvements on the existing samplers, with notable exceptions being [DN12a, BLP+13].[2] This thesis intend to fill the gap between the theory and practice of Gaussian sampling in cryptography. This is done in three parts.

Improving Existing Gaussian Samplers. We will consider the existing Gaussian samplers and try to improve them. This will be done by a statistical and geometrical analysis. In Chapter 3, we will use a divergence measure known as the *KullBack-Leibler divergence* to improve on proofs provided by the more classical statistical distance. This allows us to sample shorter vectors, which results in more secure schemes. It also shortens the required precision, which makes both existing samplers faster and easier to implement in software. In Chapter 4, we show that a large class of structured lattices (which include all the ring lattices used in efficient lattice-based cryptography) are compatible with an algorithm called the Levinson recursion. Typically, for ring lattices over some ring $\mathcal{R}$ of rank n over $\mathbb{Z}$, this enables to orthogonalize a basis of the lattice essentially $O(n)$ faster. In addition, we can use this in conjunction with Klein's sampler [Kle00, GPV08] to reduce its space complexity by a factor $O(n)$.

New Gaussian Sampler over Ring Lattices. In Chapter 5, we provide a generalization over ring lattices of the sampling algorithm of Klein's sampler. We then show that this generalized sampler can use the one from [Pei10] – known as Peikert's sampler – as a subroutine, which results in a new and hybrid sampler for ring lattices. In Chapter 6, we assess the practical interest of this sampler by comparing it with the two preexisting ones on an efficient and widely used class of lattices called NTRU lattices. Our new algorithm provides a quality-efficiency trade-off between Klein's and Peikert's sampler, and it turns out that this trade-off is very favorable to our sampler. In Chapter 7, we mix together the ideas of Fast Fourier transform and the nearest plane algorithm to obtain, for lattices over tower of rings, a nearest plane algorithm that can run as fast as quasilinear time.

Identity-Based Encryption. In Chapter 8, we use the results from the preceding chapters to provide an efficient identity-based encryption scheme (IBE) over lattices. This IBE uses Gaussian sampling as a core procedure. Compared to the state of the art in (pairing-based) IBE, our scheme does not perform as well in terms of ciphertext size, but the encryption and decryption are faster by three orders of magnitude. To conclude, Chapter 9 lists a few applications of IBE and cites a few cases where our lattice-based IBE could be particularly useful.

[2] We consider as *generic* Gaussian samplers algorithms which sample over a lattice. However, many works [Pei10, DN12a, DDLL13, Duc13, BCG+14, RVV14, Lep14, DG14] propose algorithms and improvements in the special case $\Lambda = \mathbb{Z}$.

Chapter 2

Preliminaries

"If you don't know, the thing to do is not to get scared, but to learn."

— Ayn Rand, *Atlas Shrugged*

Notations. This paragraph states the notation conventions that will be used through this document *except when stated otherwise.* Alice and Bob are metasyntactic variables which in this thesis denote (possibly interactive) participants trying to communicate securely on an insecure channel. We note $\mathbb{C}$ the set of complex numbers, $\mathbb{R}$ the set of integers, $\mathbb{Q}$ the set of rational numbers, $\mathbb{Z}$ the set of integers and $\mathbb{N}$ the set of non-negative integers. m and n will be non-negative integers such that $m \geqslant n \geqslant 0$ and we will note $H = \mathbb{K}^m$, where typically $\mathbb{K} = \mathbb{R}$ but $\mathbb{K}$ may be a number field in later sections of this thesis. Scalars will be usually noted in plain letters (such as a, b), vectors will be noted in bold letters (such as $\mathbf{a}, \mathbf{b}$) and matrices will be noted in uppercase bold letters (such as $\mathbf{A}, \mathbf{B}$). Vectors are mostly in row notation, and as a consequence vectors-matrix product is done in this order when not stated otherwise. $(a_1, ..., a_n)$ denotes the row vector formed of the a_i's, whereas $[\mathbf{a}_1, ..., \mathbf{a}_n]$ denotes the matrix whose rows are the $\mathbf{a}_i$'s.

The notation $a \triangleq b$ means that a is defined as b. A basis will either be a set $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$ of vectors linearly independent in $\mathbb{R}$ or the $n \times m$ matrix whose rows are the $\mathbf{b}_i$'s. We note $\mathrm{Span}_X(\mathbf{b}_1, ..., \mathbf{b}_k)$ (or $\mathrm{Span}_X(\mathbf{B})$) the set $\{\sum_{1 \leqslant i \leqslant k} x_i \mathbf{b}_i, x_i \in X\}$. In particular, $\mathrm{Span}_{\mathbb{Z}}(\mathbf{B})$ is an $\mathbb{Z}$-module and $\mathrm{Span}(\mathbf{B}) \triangleq \mathrm{Span}_{\mathbb{R}}(\mathbf{B})$ is a $\mathbb{R}$-vector space. $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ denote respectively the usual inner product $H \times H \to \mathbb{R}$ and the associated norm. For a scalar a, $\lfloor a \rceil$ denotes the usual rounding over a. By rounding coefficient-wise, one can extend this notation to vectors. log denotes the natural logarithm, and $\log_a$ is the logarithm in basis a.

For two real functions $f, g : \mathcal{R} \to \mathcal{R}$, we use the following asymptotic notations:

- $f = O(g)$ if and only if (iff) $\exists M > 0, x_0 \in \mathbb{R}$ such that $\forall x > x_0, |f(x)| \leqslant M|g(x)|$.
- $f = \tilde{O}(g)$ iff $f = O(g \log g)$.
- $f = o(g)$ iff $\forall \epsilon > 0, \exists x_0 \in \mathbb{R}$ such that $\forall x > x_0, |f(x)| \leqslant \epsilon |g(x)|$.
- $f = \Omega(g)$ iff $g = O(f)$.
- $f = \omega(g)$ iff $g = o(f)$.
- $f \sim g$ iff $f/x \to_\infty 1$.

2.1 Introduction to Lattices

Definition 2.1 (Lattice)**.** *Let $H = \mathbb{R}^m$. A lattice is a discrete subgroup of H. For a basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$, we note $\mathcal{L}(\mathbf{B})$ and call lattice generated by $\mathbf{B}$ the set of vectors*

$$\left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i | x_i \in \mathbb{Z} \right\}$$

A lattice is commonly noted Λ, or $\mathcal{L}(\mathbf{B})$ when provided with a basis $\mathbf{B}$.

We note that $\mathcal{L}(\mathbf{B}) = \mathrm{Span}_{\mathbb{Z}}(\mathbf{B})$. We will favor the former notation. The successive minima and determinant of a lattice are useful invariants that help to understand its geometry.

Definition 2.2 (Successive Minima of a Lattice)**.** *Let $\Lambda \subseteq \mathbb{R}^m$ be a lattice of rank n. For $i \in [\![1, n]\!]$, we note $\lambda_i(\Lambda)$ and call i-th successive minimum of Λ the following value:*

$$\lambda_i(\Lambda) = \inf_r \{\dim(\mathrm{Span}(\Lambda \cap \bar{B}(\mathbf{0}, r)) \geqslant i\}$$

Definition 2.3 (Determinant of a Lattice)**.** *Let $\Lambda = \mathcal{L}(\mathbf{B})$ be a lattice of rank n. We note $\det(\Lambda)$ and call determinant of Λ the value $\sqrt{\det(\mathbf{B}\mathbf{B}^t)}$. In the special case where $\mathbf{B}$ is a square matrix, it is immediate that $\det(\Lambda) = |\det(\mathbf{B})|$.*

One can check that definition 2.3 is independent of the basis $\mathbf{B}$ chosen, and is therefore consistent. Indeed, if two bases $\mathbf{B}_1, \mathbf{B}_2$ generate the same lattice, then $\mathbf{B}_1 = \mathbf{U}\mathbf{B}_2$ for some unimodular matrix $\mathbf{U}$. By multiplicativity of the determinant, $\mathbf{B}_1\mathbf{B}_1^t$ and $\mathbf{B}_2\mathbf{B}_2^t$ have the same determinant.

2.1.1 Problems over Lattices

For completeness, we recall the main problems over lattices. They can informally be classified in two categories. The first arise "naturally" when studying lattices.

Definition 2.4 (SVP – Shortest Vector Problem)**.** *Given a n-dimensional lattice Λ, find a lattice vector $\mathbf{v}$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.*

Definition 2.5 (ASVP_γ – Approximate Shortest Vector Problem)**.** *Given a n-dimensional lattice Λ and $\gamma \geqslant 1$ a function of n, find a lattice vector $\mathbf{v}$ such that $\|\mathbf{v}\| \leqslant \gamma \cdot \lambda_1(\Lambda)$.*

Definition 2.6 (CVP – Closest Vector Problem)**.** *Given a n-dimensional lattice Λ and a point $\mathbf{c} \in H$, find a lattice vector $\mathbf{v}$ such that $\|\mathbf{c} - \mathbf{v}\| = dist(\mathbf{c}, \lambda) \triangleq \min_{\mathbf{z}\in\Lambda} \|\mathbf{c} - \mathbf{z}\|$.*

Definition 2.7 (ACVP_γ – Approximate Closest Vector Problem)**.** *Given a n-dimensional lattice Λ and $\gamma \geqslant 1$ a function of n, find a lattice vector $\mathbf{v}$ such that $\|\mathbf{c} - \mathbf{v}\| \leqslant \gamma \cdot dist(\mathbf{c}, \Lambda)$.*

Definition 2.8 (SIVP_γ – Approximate Shortest Independent Shortest Vector Problem)**.** *Given a n-dimensional lattice Λ and $\gamma \geqslant 1$ a function of n, find n independent vectors $\mathbf{v}_i$ of Λ such that for any $i \in [\![1, n]\!]$, $\|\mathbf{v}_i\| \leqslant \gamma \cdot \lambda_i(\Lambda)$.*

Definition 2.9 (BDD_α – Bounded Distance Decoding Problem)**.** *Given a n-dimensional lattice Λ and a vector $\mathbf{t}$ such that $dist(\mathbf{t}, \mathcal{L}) < \alpha \cdot \lambda_1(\Lambda)$, find a lattice vector $\mathbf{v}$ such that $\|\mathbf{v} - \mathbf{t}\| < \alpha \cdot \lambda_1(\Lambda)$.*

The problems we just mentioned are hard on classical computers, and it is currently not known whether quantum computers can significantly speed up their resolution. This stands in contrast to problems such as the discrete logarithm and the factoring, which are known to be solvable in (probabilistic) polynomial time by a quantum computer [Sho94].

The second class of lattice problems are *ad hoc*, as they arise from cryptographic constructions. The SIS problem was introduced by Ajtai [Ajt96] and the LWE problem by Regev [Reg05].

Definition 2.10 ($\mathsf{SIS}_{n,m,q,\beta}$ – Shortest Integer Solution)**.** *Let n and $m, q = \mathsf{poly}(n)$ be integers. Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leqslant \beta$.*

Definition 2.11 ($\mathsf{LWE}_{n,m,q,\chi}$ – Learning With Errors)**.** *Let n, m, q be integers. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a uniformly random matrix, and $\mathbf{b} = \mathbf{sA} + \mathbf{e}$, where $\mathbf{e} \in \mathbb{Z}_q^m$ is sampled from a distribution χ. Recover $\mathbf{s} \in \mathbb{Z}_q^n$.*

Since SIS and LWE are *ad hoc*, their hardness may be questioned. However, a series of works showed that they are as hard as standard lattice problems [Ajt96, MR04, Reg05, BLP+13]. Under certain conditions, their ring variants (where $\mathbb{Z}$ is replaced by a ring, usually cyclotomic rings and in particular $\mathbb{Z}[x]/(x^n + 1)$ for n a power of two) were also shown [LM06, PR06, SSTX09, LPR10, LPR13b] to be as hard as variants of standard lattice problem in ideal lattices. This means that cryptography based on LWE and SIS can ultimately be made to rely of the hardness of standard lattice problems. In absence of any quantum algorithm that can solve them efficiently, lattice-based cryptography is expected to resist a potential advent of quantum computers. Cryptographic schemes that are believed to withstand this threat are regrouped under the umbrella term *post-quantum cryptography*. Other potential post-quantum areas of cryptography include code-based and hash-based cryptography.

It is to be noted that quantum computers are no longer merely theoretical and are being considered more and more seriously as a threat – although not an immediate one. Standard bodies are actively researching post-quantum resilient cryptographic solutions[1,2], and state that they will "initiate a transition to quantum resistant algorithms in the not too distant future" [NSA15].

2.1.2 Projections

Definition 2.12 (Orthogonal Complement)**.** *Let V be a subspace of H. We call orthogonal complement of V and note $V^\perp$ the set of vectors orthogonal to all the vectors of V:*

$$V^\perp = \{\mathbf{y} \in H | \forall \mathbf{x} \in V, \langle \mathbf{x}, \mathbf{y} \rangle = 0\}$$

Similarly, for a lattice $\Lambda \subseteq H$, the orthogonal lattice of Λ modulo q is:

$$\Lambda_q^\perp = \{\mathbf{y} \in H | \forall \mathbf{x} \in V, \langle \mathbf{x}, \mathbf{y} \rangle = 0 \bmod q\}$$

If $\mathbf{B}$ is a basis of V, one can verify that $V^\perp$ is the kernel of $\mathbf{x} \mapsto \mathbf{xB}^t$, so it is a $(m - \dim(V))$-dimensional subspace of H. Since $V \cap V^\perp = \mathbf{0}$, it follows that $V + V^\perp = H$ and that V and $V^\perp$ are in direct sum:

[1] http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm
[2] http://www.etsi.org/news-events/events/770-etsi-crypto-workshop-2014

Proposition 2.13. *Let V be a subspace of H. Then for any $\mathbf{x} \in H$, there exists a unique couple $(\mathbf{y}, \mathbf{z}) \in V \times V^\perp$ such that $\mathbf{x} = \mathbf{y} + \mathbf{z}$.*

Definition 2.14 (Orthogonal Projection)**.** *Let V be a subspace of H and $\mathbf{x} \in H$. We call orthogonal projection of $\mathbf{x}$ over V and note $\mathbf{Proj}(\mathbf{x}, V)$ the only vector $\mathbf{y} \in V$ verifying $(\mathbf{x} - \mathbf{y}) \perp V$.*

Existence and uniqueness of the orthogonal projection comes from Proposition 2.13. In Proposition 2.15, we give some properties of orthogonal projections that will be useful for in the rest of this work.

Proposition 2.15. *Let V be a subspace of H and $p : \mathbf{x} \mapsto \mathbf{Proj}(\mathbf{x}, V)$ be the linear map which maps a point $\mathbf{x} \in H$ onto its orthogonal projection over V. The map p verifies:*

- $p(H) = V$
- $\ker p = V^\perp$
- $p \circ p = p$
- $p \mid_V = id \mid_V$
- $\|\mathbf{x} - p(\mathbf{x})\| = \min\limits_{\mathbf{y} \in V} \|\mathbf{x} - \mathbf{y}\|$

We clarify some terminology: the set of orthogonal projections (over subspaces) of H is a subset of the more generic set of projections of H. In this thesis we only use orthogonal projections, so we do not elaborate on projections and will abusively call "the projection over V" the unique orthogonal projection over V. We now recall a definition that will be useful to compute projections.

Definition 2.16 (Moore-Penrose Pseudoinverse)**.** *Let $n \leqslant m$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ be a full-rank matrix (meaning that its rank is equal to $\min(n, m)$, which is n here). We note $\mathbf{B}^+$ and call Moore-Penrose pseudoinverse of $\mathbf{B}$ the $m \times n$ matrix uniquely defined by:*

$$\mathbf{B}^+ = \mathbf{B}^t(\mathbf{B}\mathbf{B}^t)^{-1}$$

One can check that when $n = m$, $A^+ = A^{-1}$. For this work, the most useful property of the pseudoinverse is that $\mathbf{B}\mathbf{B}^+ = \mathbf{I}_n$. Given a point $\mathbf{x} = x_0\mathbf{B} \in \mathrm{Span}(\mathbf{B})$, it allows to recover x_0 easily: $x_0 = \mathbf{x}\mathbf{B}^+$. The following proposition also gives an effective way of computing the projection over a finite subspace V.

Proposition 2.17. *Let V be a n-dimensional subspace of H and $\mathbf{B} \in \mathbb{K}^{n \times m}$ be a basis of V. For any $\mathbf{x} \in H$, the projection of $\mathbf{x}$ over V is:*

$$\mathbf{Proj}(\mathbf{x}, \mathrm{Span}(\mathbf{B})) = \mathbf{x}\mathbf{B}^+\mathbf{B} = \mathbf{x}\mathbf{B}^t(\mathbf{B}\mathbf{B}^t)^{-1}\mathbf{B}$$

In particular, if $\mathbf{y} \in \mathbb{R}^m$, then the projection of $\mathbf{x}$ over $\mathrm{Span}(\mathbf{y})$ is:

$$\mathbf{Proj}(\mathbf{x}, \mathrm{Span}(\mathbf{y})) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{y}, \mathbf{y} \rangle}\mathbf{y}$$

With the convention $\frac{\langle \mathbf{x}, \mathbf{0} \rangle}{\langle \mathbf{0}, \mathbf{0} \rangle}\mathbf{0} = \mathbf{0}$.

Proof. Let us note $p : \mathbf{x} \in H \mapsto \mathbf{x}\mathbf{B}^+\mathbf{B}$. It is obvious that $p(\mathbf{x}) = (\mathbf{x}\mathbf{B}^+)\,\mathbf{B} \in V$. On the other hand,

$$(\mathbf{x} - p(\mathbf{x}))\mathbf{B}^t = \mathbf{x}\mathbf{B}^t - \mathbf{x}\underbrace{\mathbf{B}^+\mathbf{B}\mathbf{B}^t}_{=\mathbf{B}^t} = \mathbf{0}$$

The result then follows from the definition of the projection. □

Two vectors $\mathbf{x}, \mathbf{y}$ are said to be orthogonal if their inner product is zero. A basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ is said to be orthogonal if its vectors are pairwise orthogonal: $\forall i \neq j, \langle \mathbf{b}_i, \mathbf{b}_j \rangle = 0$. If in addition they are all unit vectors (meaning that their norm is 1), then $\mathbf{B}$ is said to be an orthonormal basis.

Orthogonal and orthonormal bases are very useful: for a basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ of some subspace V and a point $\mathbf{x} \in V$, they allow to compute a decomposition of $\mathbf{x}$ over the vectors of $\mathbf{B}$ in a parallelizable and efficient way:

- If $\mathbf{B}$ is orthogonal, then $\mathbf{x} = \sum_{i \in [\![1,n]\!]} \frac{\langle \mathbf{x}, \mathbf{b}_i \rangle}{\|\mathbf{b}_i\|^2} \mathbf{b}_i$
- If $\mathbf{B}$ is orthonormal, then $\mathbf{x} = \sum_{i \in [\![1,n]\!]} \langle \mathbf{x}, \mathbf{b}_i \rangle \mathbf{b}_i$

2.1.3 The Gram-Schmidt Orthogonalization

As we just said, orthogonal bases are very useful to decompose a vector over a basis, which leads to countless applications inside and outside the scope of this thesis. Fot this reason, it is often desirable, given a basis $\mathbf{B}$, to have an orthogonal basis $\tilde{\mathbf{B}}$ that generates the same space as $\mathbf{B}$. An infinite number of such bases exist, but for lattices we are interested in a very specific one described in the present section. We add the following notation: for a basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and $k \leqslant n$, we note $\mathbf{B}_k \triangleq \{\mathbf{b}_1, ..., \mathbf{b}_k\}$.

Lemma 2.18. *Let $H = \mathbb{R}^m$ and $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$ be a basis. For any $k \in [\![1, n]\!]$, we note $V_k \triangleq \text{Span}(\mathbf{B}_k)$. There is a unique basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\} \in H^n$ verifying any of these equivalent properties:*

1. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \mathbf{Proj}(\mathbf{b}_k, V_{k-1})$
2. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{b}_k, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \tilde{\mathbf{b}}_j$
3. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k \perp V_{k-1}$ *and* $(\mathbf{b}_k - \tilde{\mathbf{b}}_k) \in V_{k-1}$

Noting $\tilde{V}_k \triangleq \text{Span}(\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_k)$, we also have: $\forall k \in [\![1, n]\!], \tilde{V}_k = V_k$.

Proof. We first prove the equivalence of the conditions 1, 2 and 3. They are equivalent at step 1. We suppose it is the case up to step $k-1$ and prove the equivalence at step k:

- $\boxed{1 \Leftrightarrow 2}$ First let us notice that $V_{k-1} = \tilde{V}_{k-1}$ (see condition 2). Observing that for any $j < k$, the $\tilde{\mathbf{b}}_j$'s are pairwise orthogonals (see condition 3), we get for any $\mathbf{v} \in H$:

$$\mathbf{Proj}(\mathbf{v}, V_{k-1}) = \mathbf{Proj}(\mathbf{v}, \tilde{V}_{k-1}) = \sum_{j=1}^{k-1} \frac{\langle \mathbf{v}, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle} \tilde{\mathbf{b}}_j$$

  Where the second equality comes from the first equality of Proposition 2.17.

- $\boxed{2 \Rightarrow 3}$ $\tilde{\mathbf{b}}_k \perp \tilde{V}_{k-1}$ is a simple computation, and $(\mathbf{b}_k - \tilde{\mathbf{b}}_k) \in V_{k-1}$ is straightforward.

- $\boxed{3 \Rightarrow 1}$ We proved that $(1 \Rightarrow 3)$, so $\mathbf{b}_k = \mathbf{Proj}(\mathbf{b}_k, V_{k-1}) + (\mathbf{b}_k - \mathbf{Proj}(\mathbf{b}_k, V_{k-1}))$ yields a decomposition of $\mathbf{b}_k$ over V_{k-1} and $V_{k-1}^{\perp}$. Since V_{k-1} and $V_{k-1}^{\perp}$ are in direct sum, such a decomposition is unique.

The existence and uniqueness of $\tilde{\mathbf{B}}$ come from the deterministic formula in condition 2, as does the fact that $V_k = \tilde{V}_k$. □

We now define the Gram-Schmidt Orthogonalization.

Definition 2.19 (Gram-Schmidt Orthogonalization)**.** *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$ *be a basis. We call Gram-Schmidt orthogonalization (or GSO) of* $\mathbf{B}$ *and note* $\tilde{\mathbf{B}}$ *the unique set* $\{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\} \in H^n$ *verifying one of the equivalent properties of Lemma 2.18. When clear from context, we also note* $\tilde{\mathbf{b}}_i$ *the* i*-th vector of* $\tilde{\mathbf{B}}$*, which is also the orthogonalization of* $\mathbf{b}_i$ *with respect to the previous vectors* $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.

We have provided three equivalent definitions of the Gram-Schmidt orthogonalization in Lemma 2.18. The first one is geometrical, the second one is a mathematical formula, and the third one looks at each vector of the basis as its decomposition over two orthogonal vector spaces. Although the two first definitions are standard and useful for comprehension and computation, the third one is less common and we will mostly use it in proofs, to show that a basis is indeed the GSO of another one. The second notations can easily be translated into a constructive algorithm for the GSO, as shown by Algorithm 2.1.

Algorithm 2.1 GramSchmidtProcess($\mathbf{B}$)

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$
Ensure: Gram-Schmidt reduced basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$
1: **for** $i = 1, ..., n$ **do**
2: $\quad \tilde{\mathbf{b}}_i \leftarrow \mathbf{b}_i$
3: $\quad$ **for** $j = 1, ..., i-1$ **do**
4: $\quad\quad L_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2}$
5: $\quad\quad \tilde{\mathbf{b}}_i \leftarrow \tilde{\mathbf{b}}_i - L_{i,j}\tilde{\mathbf{b}}_j$
6: $\quad$ **end for**
7: **end for**
8: **return** $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

Gram-Schmidt orthogonalization also implicitly defines a lower triangular matrix $\mathbf{L} = (L_{i,j})_{1 \leqslant i,j \leqslant n}$ which is also the unique matrix verifying $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$. This matrix is used in lattice-based cryptography but also has applications beyond its scope. Faster algorithms for computing $\tilde{\mathbf{B}}$ and $\mathbf{L}$ when the basis $\mathbf{B}$ is geometrically structured are given in Chapter 4. The Gram-Schmidt orthogonalization also helps to compute or bound lattice invariants.

Proposition 2.20. *Let* $\Lambda = \mathcal{L}(\mathbf{B})$ *be a lattice of rank* n.

- $\det(\Lambda) = \prod_i \|\tilde{\mathbf{b}}_i\| = \det(\mathcal{L}(\tilde{\mathbf{B}}))$
- $\min_{j \geqslant i} \|\tilde{\mathbf{b}}_j\| \leqslant \lambda_i(\mathcal{L}(\mathbf{B})) \leqslant \max_{j \leqslant i} \|\tilde{\mathbf{b}}_j\|$

2.2 Babai's Algorithms

Given a lattice $\mathcal{L}(\mathbf{B})$ and a point $\mathbf{c}$, Babai's algorithms are used to find a point $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ close to $\mathbf{c}$. Although they were known before,[3] Babai was the first to formalize them in [Bab85, Bab86].

Both algorithms are deterministic and give approximate solutions to classical lattice problems related to CVP. Before giving the algorithms, we recall some definitions that will be useful to analyze them.

[3]The round-off algorithm is used in [Len82], and the Nearest Plane in [LLL82].

Definition 2.21 (Gram-Schmidt Norm)**.** *Let* $\mathbf{B} = (\mathbf{b}_i)_{i \in I}$ *be a finite basis, and* $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_i)_{i \in I}$ *its Gram-Schmidt orthogonalization. The Gram-Schmidt norm of* $\mathbf{B}$ *is the value*

$$|\tilde{\mathbf{B}}| = \max_{i \in I} \|\tilde{\mathbf{b}}_i\|$$

Definition 2.22 (Singular Value Decomposition)**.** *Let* $\mathbb{K}$ *be either the field of real or complex numbers, and* $\mathbf{B} \in \mathbb{K}^{n \times m}$ *be a matrix.* $\mathbf{B}$ *can be decomposed as follows:*

$$\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^\star$$

Where $\mathbf{U} \in \mathbb{K}^{n \times n}$ *and* $\mathbf{V} \in \mathbb{K}^{m \times m}$ *are unitary matrices,* $\Sigma \in \mathbb{K}^{n \times m}$ *is a diagonal matrix with real non-negative coefficients, and* $\mathbf{V}^\star$ *denotes the conjugate transpose matrix of* $\mathbf{V}$*. The diagonal entries of* Σ *are known as the singular values of* $\mathbf{B}$*, and commonly noted* $s_1(\mathbf{B}), ..., s_n(\mathbf{B})$ *in decreasing order.*

Definition 2.23 (Spectral Norm)**.** *Let* $\mathbf{B} \in \mathbb{R}^{n \times m}$ *be a matrix. The spectral norm of* $\mathbf{B}$ *is the value*

$$\max_{\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\|\mathbf{x}\mathbf{B}\|}{\|\mathbf{x}\|}$$

One can check that this definition coincides with the *operator norm* induced on the space of matrices of $\mathbb{R}^{n \times m}$ by the euclidean norm. It also coincides with the maximal singular value $s_1(\mathbf{B})$ of $\mathbf{B}$. The literature about discrete Gaussian sampling use mostly the notation $s_1(\mathbf{B})$, so we will keep this notation for the spectral norm.

Definition 2.24 (Fundamental Parallelepiped of a Basis)**.** *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ *be a basis. We note* $\mathcal{P}(\mathbf{B})$ *and call fundamental parallelepiped generated by* $\mathbf{B}$ *the set*

$$\left\{ \sum_i x_i \mathbf{b}_i | x_i \in \left[-\frac{1}{2}, \frac{1}{2} \right] \right\} = \mathrm{Span}_{\left[-\frac{1}{2}, \frac{1}{2}\right]}(\mathbf{B})$$

We now give the definition of the round-off algorithm, which provides a simple way, given a point $\mathbf{c}$ and a short basis $\mathbf{B}$, to find a "reasonably close" point of $\mathcal{L}(\mathbf{B})$ that is close to $\mathbf{c}$. We recall that $\lfloor \cdot \rceil$ denotes the coefficient-wise rounding in $\mathbb{Z}$ of a vector.

Algorithm 2.2 RoundOff$(\mathbf{B}, \mathbf{c})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathbb{Z}^{n \times m}$ of a lattice Λ, target $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$
Ensure: $\mathbf{v} \in \mathcal{L}(\mathbf{B})$
1: $\mathbf{v} \leftarrow \lfloor \mathbf{c}\mathbf{B}^+ \rceil \mathbf{B}$
2: **return** $\mathbf{v}$

Proposition 2.25. *Let* $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$*. Algorithm 2.2 outputs a point* $\mathbf{v} \in \Lambda$ *such that*

$$\mathbf{c} - \mathbf{v} \in \mathcal{P}(\mathbf{B})$$

It follows that $\|\mathbf{c} - \mathbf{v}\| \leqslant \frac{\sqrt{n}}{2} \cdot s_1(\mathbf{B})$*.*

Proof. The fact that $\mathbf{v} \in \Lambda$ is immediate from its definition. We now prove the bound over $\mathbf{c} - \mathbf{v}$. Let $\mathbf{c} = \mathbf{c}_0\mathbf{B}$ for some $\mathbf{c}_0 \in \mathbb{R}^n$. Then $\mathbf{c}_0 = \mathbf{c}\mathbf{B}^+$ (since $\mathbf{B}\mathbf{B}^+ = \mathbf{I}_n$) and we have:

$$\mathbf{c} - \mathbf{v} = \mathbf{c}_0\mathbf{B} - \lfloor \mathbf{c}\mathbf{B}^+ \rceil \mathbf{B} = \underbrace{(\mathbf{c}_0 - \lfloor \mathbf{c}_0 \rceil)}_{\in \left[-\frac{1}{2}, \frac{1}{2}\right]^n} \mathbf{B}$$

□

Following the notations of Proposition 2.25, one can notice that Algorithm 2.2 outputs the same result for any input in $\mathbf{c} + \mathrm{Span}(\mathbf{B})^{\perp}$.

Algorithm 2.3 NearestPlane$(\mathbf{B}, \mathbf{c})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathbb{Z}^{n \times m}$, its GSO $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$, target $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$
Ensure: $\mathbf{v} \in \mathcal{L}(\mathbf{B})$

$\mathbf{c}_n \leftarrow \mathbf{c}$
$\mathbf{v}_n \leftarrow \mathbf{0}$
for $i \leftarrow n, ..., 1$ **do**
  $d_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$
  $z_i \leftarrow \lfloor d_i \rceil$
  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \mathbf{b}_i$
  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \mathbf{b}_i$
end for
return $\mathbf{v}_0$

Proposition 2.26. *Let* $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$. *Algorithm 2.3 outputs a point* $\mathbf{v} \in \Lambda$ *such that*

$$\mathbf{c} - \mathbf{v} \in \mathcal{P}(\tilde{\mathbf{B}})$$

It follows that $\|\mathbf{c} - \mathbf{v}\|^2 \leqslant \frac{1}{4} \sum_{i=1}^{n} \|\tilde{\mathbf{b}}_i\|^2 \leqslant \frac{n}{4} |\tilde{\mathbf{B}}|^2$.

Proof. The fact that $\mathbf{v} \in \Lambda$ follows from the fact that $\mathbf{c}_0 = \sum_i z_i \mathbf{b}_i$ for some $(z_i)_{i \in [\![1,n]\!]} \in \mathbb{Z}^n$. We now prove the bound over $\mathbf{c} - \mathbf{v}$ by showing that at for any $i \in [\![0, n]\!]$, the following condition holds:

$$\forall j \in [\![i+1, n]\!], \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \tag{2.1}$$

Indeed, $\mathbf{c}_0 \in \mathrm{Span}(\tilde{\mathbf{B}})$ and $\tilde{\mathbf{B}}$ is orthogonal so $\mathbf{c}_0 = \sum_i \frac{\langle \mathbf{c}_0, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j$. It is then obvious that having condition (2.1) hold at $i = 0$ is equivalent to proving the bound over $\mathbf{c} - \mathbf{v}$ since $\mathbf{c}_0 = \mathbf{c} - \mathbf{v}$. We show by descending induction over i that condition (2.1) holds for any $i \in [\![0, n]\!]$.

For $i = n$, the set $[\![i+1, n]\!]$ is empty so condition (2.1) is a tautology. If condition (2.1) is true at step i, then from $\mathbf{c}_{i-1} \triangleq \mathbf{c}_i - \left\lfloor \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\|^2} \right\rceil \mathbf{b}_i$ it is easy to check that:

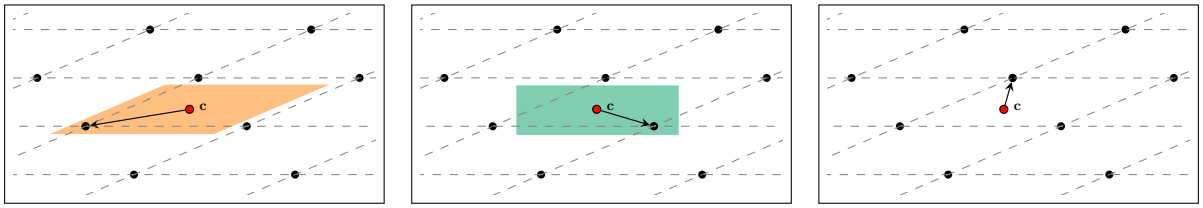
- For $j > i$, $\frac{\langle \mathbf{c}_{i-1}, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} = \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2}$ since $\tilde{\mathbf{b}}_j$ is orthogonal to $\mathbf{b}_i$, so adding a scalar multiple of $\mathbf{b}_i$ to $\mathbf{c}_i$ doesn't change its dot product with $\tilde{\mathbf{b}}_j$.
- For $j = i$, $\frac{\langle \mathbf{c}_{i-1}, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} = \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} - \left\lfloor \frac{\langle \mathbf{c}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \right\rceil \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ using the fact that $\langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle = \|\tilde{\mathbf{b}}_j\|^2$

By induction, condition (2.1) holds as long as $\mathbf{c}_i$ and the $\tilde{\mathbf{b}}_j$'s are defined, which is all the way down to $i = 0$. This concludes the proof.

The bound over $\|\mathbf{c} - \mathbf{v}\|^2$ is then given by the Pythagorean theorem. □

2.2.1 Comparing the Round-Off and Nearest Plane Algorithms

If we consider elementary operations in $\mathbb{R}$ to be of complexity $O(1)$, then both RoundOff and NearestPlane have a $O(nm)$ complexity. They are deterministic and the qualities of their outputs depend on the geometry of $\mathbf{B}$ (for RoundOff) and of $\tilde{\mathbf{B}}$ (for NearestPlane), as stated by propositions 2.25 and 2.26. However, even though $\mathcal{P}(\mathbf{B})$ and $\mathcal{P}(\tilde{\mathbf{B}})$ have the same volume, $\tilde{\mathbf{B}}$ is orthogonal so $\mathcal{P}(\tilde{\mathbf{B}})$ may not contain vectors as large as $\mathcal{P}(\mathbf{B})$ do. A simple basis on which NearestPlane can perform arbitrarily better than RoundOff is $\mathbf{B} = \{[0, 1], [1, n]\}$ when $n \to +\infty$. Some limitations of both algorithms are given in Lemma 2.27 and illustrated in Figure 2.1. In a nutshell, the nearest-plane algorithm outputs closer vectors than the round-off algorithm in the worst and average cases, but we can find counterexamples where the round-off algorithm returns a closer vector for specific bases and target points.



RoundOff (2.2) NearestPlane (2.3) Actual closest vector

In the worst and average cases, the nearest plane algorithm performs better than the round-off algorithm. However, it may not always find a closest vector to c, even in small dimension.

Figure 2.1: Comparing Babai's algorithms.

Lemma 2.27. *The following properties hold:*

1. *For any fixed basis* $\mathbf{B}$*, the nearest plane algorithm outputs closer vectors than the round-off algorithm in the worst case:*

$$\max_{\mathbf{c}\in\mathrm{Span}(\mathbf{B})} \|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\| \leqslant \max_{\mathbf{c}\in\mathrm{Span}(\mathbf{B})} \|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|$$

2. *For any fixed basis* $\mathbf{B}$*, the nearest plane algorithm outputs closer vectors than the round-off algorithm on average:*

$$\mathbb{E}_{\mathbf{c}\xleftarrow{\$}\mathrm{Span}(\mathbf{B})/\mathcal{L}(\mathbf{B})}\left[\|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\|^2\right] \leqslant \mathbb{E}_{\mathbf{c}\xleftarrow{\$}\mathrm{Span}(\mathbf{B})/\mathcal{L}(\mathbf{B})}\left[\|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|^2\right]$$

3. *The nearest plane algorithm does not always output closer lattice points than the round-off algorithm. More specifically, there exist a basis* $\mathbf{B}$ *and a point* $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$ *such that:*

$$\|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\| > \|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|$$

Proof. We prove the three propositions separately:

1. Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ be a basis and $\mathbf{c} = \frac{1}{2}\sum_{i\in[\![1,n]\!]} \tilde{\mathbf{b}}_i$. As $\mathbf{c} \in \mathcal{P}(\tilde{\mathbf{B}})$, $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})$ outputs the point $\mathbf{v} = \mathbf{0}$ and $\|\mathbf{c} - \mathbf{v}\|$ then reaches the upper bound of Proposition 2.26 for the nearest plane algorithm.[4]

[4]Depending on the rounding conventions, the nearest plane might output another point, but the conclusion regarding the bound $\|\mathbf{c} - \mathbf{v}\|$ would be the same.

We now construct $\mathbf{d}$ such that $\|\mathbf{d} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{d})\| \geqslant \|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\|$. We proceed as follows: let $\mathbf{d}_n = \frac{1}{2}\mathbf{b}_n$ and for any $i < n$, $\mathbf{d}_i = \mathbf{d}_{i+1} + \frac{(-1)^{k_i}}{2}\mathbf{b}_i$ where $k_i = 0$ if $\langle \mathbf{d}_{i+1}, \tilde{\mathbf{b}}_i \rangle > 0$, $k_i = 1$ otherwise. The point $\mathbf{d} = \mathbf{d}_1$ verifies:

$$\forall i, \left| \frac{\langle \mathbf{d}, \tilde{\mathbf{b}}_i \rangle}{\langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_i \rangle} \right| \geqslant \frac{1}{2} \tag{2.2}$$

As $\mathbf{d}$ is in $\mathcal{P}(\mathbf{B})$, applying $\mathsf{RoundOff}(\mathbf{B}, \mathbf{d})$ outputs $\mathbf{0}$. From equation 2.2, it is immediate that $\|\mathbf{d} - \mathbf{0}\| \geqslant \|\mathbf{c} - \mathbf{0}\|$. This yields a constructive proof that in the worst case, the round-off algorithm outputs further vectors than the nearest plane algorithm.

2. We first notice that for any fixed $\mathbf{c}$ and any $\mathbf{c}' \in \mathbf{c} + \mathcal{L}(\mathbf{B})$:

$$\|\mathbf{c}' - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c}')\| = \|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\| \tag{2.3}$$

Which makes it consistent to define $\mathbb{E}_{\mathbf{c} \xleftarrow{\$} \mathrm{Span}(\mathbf{B})/\mathcal{L}(\mathbf{B})} [\|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|^2]$. Now let $\mathcal{P}'(\mathbf{B}) \triangleq \mathrm{Span}_{(-\frac{1}{2}, \frac{1}{2}]}(\mathbf{B})$. We have:

$$\begin{aligned}
\mathbb{E}_{\mathbf{c} \xleftarrow{\$} \mathrm{Span}(\mathbf{B})/\mathcal{L}(\mathbf{B})} [\|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|^2] &= \mathbb{E}_{\mathbf{c} \xleftarrow{\$} \mathcal{P}'(\mathbf{B})} [\|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|^2] \\
&= \frac{1}{\mathrm{Vol}(\mathcal{P}'(\mathbf{B}))} \int_{\mathcal{P}'(\mathbf{B})} \|\mathbf{c}\|^2 d^n\mathbf{c} \\
&= \frac{1}{\mathrm{Vol}(\mathcal{P}(\mathbf{B}))} \int_{\mathcal{P}(\mathbf{B})} \|\mathbf{c}\|^2 d^n\mathbf{c} \\
&= \int_{[-\frac{1}{2}, \frac{1}{2}]^n} \|\mathbf{x}\mathbf{B}\|^2 d^n\mathbf{x}
\end{aligned}$$

Hereinabove, the first equality holds because of equation 2.3. The second one is straightforward once we notice that $\mathsf{RoundOff}(\mathbf{B}, \mathbf{c}) = \mathbf{0}$ for $\mathbf{c}$ in the integration domain. The third one notices that the measure of $\mathcal{P}(\mathbf{B}) \backslash \mathcal{P}'(\mathbf{B})$ is zero, therefore the integral of $\|\mathbf{c}\|^2$ over $\mathcal{P}'(\mathbf{B})$ or $\mathcal{P}(\mathbf{B})$ is the same. For the fourth one, we use the change of variable $\mathbf{c} = \mathbf{x}\mathbf{B}$, where $\mathbf{x} = (x_i)_{i \in [\![1,n]\!]}$. Similarly, we show that

$$\mathbb{E}_{\mathbf{c} \xleftarrow{\$} \mathrm{Span}(\mathbf{B})/\mathcal{L}(\mathbf{B})} \left[\|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\|^2 \right] = \int_{[-\frac{1}{2}, \frac{1}{2}]^n} \|\mathbf{x}\tilde{\mathbf{B}}\|^2 d^n\mathbf{x}$$

To conclude the proof, we now show by induction that for any $k \in [\![1, n]\!]$, if we note $\mathbf{x}_k \triangleq (x_i)_{i \in [\![1,k]\!]}$ and $\mathbf{B}_k \triangleq \{\mathbf{b}_1 ... \mathbf{b}_k\}$, we have:

$$\int_{[-\frac{1}{2}, \frac{1}{2}]^k} \|\mathbf{x}_k \mathbf{B}_k\|^2 d^k\mathbf{x}_k \geqslant \int_{[-\frac{1}{2}, \frac{1}{2}]^k} \|\mathbf{x}_k \tilde{\mathbf{B}}_k\|^2 d^k\mathbf{x}_k \tag{2.4}$$

The inequality is trivial (and is an equality) for $k = 1$. Suppose it is true up to a given k, we have:

$$\int_{[-\frac{1}{2}, \frac{1}{2}]^{k+1}} \|\mathbf{x}_{k+1} \tilde{\mathbf{B}}_{k+1}\|^2 d^{k+1}\mathbf{x}_{k+1} = \int_{[-\frac{1}{2}, \frac{1}{2}]^k} \|\mathbf{x}_k \tilde{\mathbf{B}}_k\|^2 d^k\mathbf{x}_k + \int_{[-\frac{1}{2}, \frac{1}{2}]} \|x_{k+1} \tilde{\mathbf{b}}_{k+1}\|^2 dx_{k+1} \tag{2.5}$$

On the other hand, the decomposition $\mathbf{x}_{k+1}\mathbf{B}_{k+1} = \mathbf{x}_k\mathbf{B}_k + x_{k+1}\mathbf{b}_{k+1}$ yields:

$$\begin{aligned}
\int_{[-\frac{1}{2}, \frac{1}{2}]^{k+1}} \|\mathbf{x}_{k+1}\mathbf{B}_{k+1}\|^2 d^{k+1}\mathbf{x}_{k+1} &= \int_{[-\frac{1}{2}, \frac{1}{2}]^k} \|\mathbf{x}_k\mathbf{B}_k\|^2 d^k\mathbf{x}_k + \int_{[-\frac{1}{2}, \frac{1}{2}]} \|x_{k+1}\mathbf{b}_{k+1}\|^2 dx_{k+1} \\
&\quad + 2 \int_{[-\frac{1}{2}, \frac{1}{2}]^{k+1}} \langle \mathbf{x}_k\mathbf{B}_k, x_{k+1}\mathbf{b}_{k+1} \rangle d^{k+1}\mathbf{x}_{k+1}
\end{aligned} \tag{2.6}$$

From the fact that the dot product is an odd function in each term (*ie* $\langle x, -y\rangle = \langle -x, y\rangle = -\langle x, y\rangle$), the last integral in equation 2.5 is zero. This leaves two nonzero terms in the right member of equation 2.5: the first one is larger than $\int_{[-\frac{1}{2},\frac{1}{2}]^k} \|\mathbf{x}_k\tilde{\mathbf{B}}_k\|^2 d^k\mathbf{x}_k$ by induction hypothesis and the second one is larger than $\int_{[-\frac{1}{2},\frac{1}{2}]} \|x_{k+1}\tilde{\mathbf{b}}_{k+1}\|^2 dx_{k+1}$ since $\|\mathbf{b}_{k+1}\| \geqslant \|\tilde{\mathbf{b}}_{k+1}\|$. This implies equation 2.4 at step $k+1$ and therefore concludes the proof.

3. Let $\epsilon \in (0, 1/6)$ and

$$\mathbf{B} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ \epsilon & 1 & 0 \\ 1 & -1/\epsilon & 1 \end{bmatrix}}_{\mathbf{L}} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\tilde{\mathbf{B}}} = \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & \epsilon & 0 \\ 1 & -1 & 1 \end{bmatrix}$$

Let $\mathbf{c} = \epsilon\mathbf{b}_1 + \mathbf{b}_2 + \epsilon\mathbf{b}_3 = 3\epsilon\tilde{\mathbf{b}}_1 + \epsilon\tilde{\mathbf{b}}_3$. We have $\mathsf{RoundOff}(\mathbf{B}, \mathbf{c}) = \mathbf{b}_2$ and $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{c}) = 0$, so that

$$\|\mathbf{c} - \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})\|^2 = 10\epsilon^2 > 6\epsilon^2 = \|\mathbf{c} - \mathsf{RoundOff}(\mathbf{B}, \mathbf{c})\|^2$$

□

2.2.2 Uses of the Nearest Plane Algorithm

One of the most notable uses of the nearest plane algorithm is in the LLL algorithm [LLL82] (which ironically, was invented before Babai published [Bab85, Bab86]). More precisely, it is an essential component of the size-reduction procedure. We recall that for $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and $k \leqslant n$, we note $\mathbf{B}_k \triangleq \{\mathbf{b}_1, ..., \mathbf{b}_k\}$. The size-reduction can very simply described using the round-off algorithm:

Algorithm 2.4 $\mathsf{SizeReduce}(\mathbf{B})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathbb{Z}^{n\times m}$
Ensure: Size-reduction of $\mathbf{B}$
1: **for** $k \leftarrow 2, ..., n$ **do**
2: $\quad \mathbf{b}_k \leftarrow \mathbf{b}_k - \mathsf{NearestPlane}(\mathbf{B}_{k-1}, \mathbf{b}_k)$
3: **end for**
4: **return** $\mathbf{B}$

And the LLL algorithm can be defined using only the size-reduction and the orthogonalization of a basis.

Algorithm 2.5 $\mathsf{LLL}(\mathbf{B}, \delta)$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathbb{Z}^{n\times m}$
Ensure: δ-LLL-reduction of $\mathbf{B}$
1: $\mathsf{SizeReduce}(\mathbf{B})$
2: **if** $\delta\|\tilde{\mathbf{b}}_k\|^2 > \|\tilde{\mathbf{b}}_{k+1}\|^2 + \frac{\langle \mathbf{b}_{k+1}, \tilde{\mathbf{b}}_k\rangle^2}{\|\tilde{\mathbf{b}}_k\|^2}$ for some k **then**
3: $\quad \mathbf{b}_k \leftrightarrow \mathbf{b}_{k+1}$
4: $\quad$ **return** $\mathsf{LLL}(\mathbf{B})$
5: **else**
6: $\quad$ **return** $\mathbf{B}$
7: **end if**

The LLL algorithm has numerous applications in cryptanalysis but also beyond (see [NV10]). As this algorithm is outside the scope of this thesis, we do not elaborate on it. Nevertheless, it would be interesting to see if the techniques we develop in this thesis could benefit to it. Given a point $\mathbf{c} \in \mathrm{Span}(\mathbf{B})$, NearestPlane can also be used to solve CVP-related problems over $\Lambda(\mathbf{B})$ and $\mathbf{c}$:

- For any lattice $\Lambda \subset \mathbb{R}^n$ of dimension n, there exists a basis $\mathbf{B}$ with which NearestPlane solves $\mathsf{BDD}_{\frac{1}{2n}}$.
- One can show that NearestPlane solves ACVP_γ for $\gamma \geqslant \sqrt{n} \cdot \frac{\max_i \|\tilde{\mathbf{b}}_i\|}{\min_i \|\tilde{\mathbf{b}}_i\|}$.
- If the basis $\mathbf{B}$ is δ-LLL-reduced with parameter $\delta = \frac{3}{4}$, one can show that NearestPlane solves ACVP_γ for $\gamma = 2^{n/2}$ [Bab85, Bab86]. While of theoretic interest, this value of γ is in practice often much worse than the previous bound $\sqrt{n} \cdot \frac{\max_i \|\tilde{\mathbf{b}}_i\|}{\min_i \|\tilde{\mathbf{b}}_i\|}$.

2.2.3 Using Babai's Algorithms in Public Key Cryptography

Given a short basis $\mathbf{B}$, Babai's algorithms can give, for any point $\mathbf{c} \leftarrow H(\mathbf{m})$, a lattice point that is close to $\mathbf{c}$ in a reasonable time. In addition, it is in general hard to find a short basis from a long basis. With these fact in mind, it is tempting to devise a public key scheme where $\mathbf{B}$ would be a private key using Babai's algorithms to answer queries. And indeed, this was attempted by Goldwasser, Goldreich and Halevi with the GGH signature scheme [GGH97], as well as Hoffstein, Howgrave-Graham, Pipher, Silverman and Whyte with the NTRUSign scheme [HHGP+03]. In essence, the idea was the following:

- Sign : Hash a message $\mathbf{m}$ into a point $\mathbf{c}$ of the space and use $\mathsf{RoundOff}(\mathbf{B}, \mathbf{c})$ to find a lattice point $\mathbf{s} \in \Lambda$ close to $\mathbf{c}$. Output $\mathbf{s}$.
- Verify : Accept iff $\mathbf{s} \in \Lambda$ and $\|\mathbf{s} - \mathbf{c}\|$ is small.

However, this idea fell short due to the fact that the signature $\mathbf{s}$ was not chosen depending only of its closeness to $\mathbf{c}$ but also of the geometry of the *private key*. This is obvious in Propositions 2.25, 2.26 as well as in Figure 2.1.

Nguyen and Regev exploited this leakage of the private key to perform a total break of the schemes [NR06] with a statistical attack that can be synthesized in the ulterior Figure 2.7. Perturbations countermeasures were proposed by the authors of [HHGP+03] but their security was questioned in [Wan10] and they were subsequently broken by Ducas and Nguyen [DN12b].

2.3 Discrete Gaussians

The point of failure in GGH and NTRUSign came from the fact that the signature depended of the geometry of the basis $\mathbf{B}$ used for signing. We carefully note that the geometry of the lattice and the geometry of the basis are two different things. The first one is a lattice invariant, but two bases can have different geometries. In a hash-and-sign scheme *à la* [GGH97, HHGP+03], a signature necessarily depends on the lattice, since it is a lattice point. However, it doesn't inherently have to depend of the basis. As an example, suppose the signer is able to return the closest vector to any point. Then clearly the signature would not depend on any basis.

Of course, returning the closest vector is NP-hard in general. However, returning a point independent of the basis would be sufficient to thwart the aforementioned attacks. In 2008, Gentry, Peikert and Vaikuntanathan [GPV08] showed how to return a point that would not depend on the basis used. Instead of taking as a signature the intersection of $H(\mathbf{m}) + \mathcal{P}(\mathbf{B})$ and of the lattice $\mathcal{L}(\mathbf{B})$ (which is in essence what GGH and the original NTRUSign did), [GPV08] sampled it from a spherical *discrete Gaussian distribution*. This distribution requires some additional definitions which are given in this section.

2.3.1 The Statistical Distance

The statistical distance is a prevalent tool in cryptography and is used to measure the "closeness" of two distributions.

Definition 2.28 (Statistical Distance)**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions over a common countable set Ω. The statistical distance – also known as total variation distance – between $\mathcal{P}$ and $\mathcal{Q}$ is noted $\Delta(\mathcal{P}, \mathcal{Q})$ and defined as:*

$$\Delta(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{i \in \Omega} |\mathcal{P}(i) - \mathcal{Q}(i)|$$

One can check that Δ is indeed a distance. For cryptography, it is an interesting tool as it gives bounds on the indistinguishability of two distributions.

Proposition 2.29. *For any distributions over which it is defined, the statistical distance verifies:*

- *Sub-additivity:* $\Delta(\mathcal{P}_0 \times \mathcal{P}_1, \mathcal{Q}_0 \times \mathcal{Q}_1) = \Delta(\mathcal{P}_0, \mathcal{Q}_0) + \Delta(\mathcal{P}_1, \mathcal{Q}_1)$
- *Preservation under any transformation: for any function f, the following inequality holds and is an equality when f is injective over the support of $\mathcal{P}$ and $\mathcal{Q}$:*

$$\Delta(f(\mathcal{P}), f(\mathcal{Q})) \leqslant \Delta(\mathcal{P}, \mathcal{Q})$$

The preservation property is very useful to assess the security of a scheme: replacing f by an algorithm that takes as input a distribution $\mathcal{P}_{\mathcal{S}}$ depending of a scheme $\mathcal{S}$ and outputs 1 if it breaks the scheme $\mathcal{S}$, it allows one to argue that if f fails at breaking $\mathcal{S}$, then it also fails at breaking another scheme $\mathcal{T}$ as long as $\Delta(\mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{T}})$ is small.

2.3.2 Gaussians

The Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as follows:

$$\rho(\mathbf{x}) \triangleq \exp\left(-\|\mathbf{x}\|^2/2\right)$$

If $\mathbf{B} \in \mathbb{R}^{n \times n}$ is a nonsingular matrix, and $\mathbf{c} \in \mathbb{R}^n$, then we extend this definition:

$$\rho_{\mathbf{B},\mathbf{c}}(\mathbf{x}) \triangleq \rho\left((\mathbf{x} - \mathbf{c})\mathbf{B}^{-1}\right)$$

Let $\Sigma = \mathbf{B}^t\mathbf{B}$. Then for any unitary matrix $\mathbf{U}$, $\mathbf{B}^t\mathbf{B} = (\mathbf{UB})^t(\mathbf{UB}) = \Sigma$. We therefore also use the notation $\rho_{\sqrt{\Sigma},\mathbf{c}}$ for $\rho_{\mathbf{B},\mathbf{c}}$, where $\sqrt{\Sigma}$ is an arbitrary matrix verifying $\sqrt{\Sigma}^t\sqrt{\Sigma} = \Sigma$.

For a lattice Λ, we note $\rho_{\mathbf{B},\mathbf{c}}(\Lambda) \triangleq \sum_{\mathbf{x}\in\Lambda} \rho_{\mathbf{B},\mathbf{c}}(\mathbf{x})$. We can now define the discrete Gaussian distribution $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$ over Λ:

$$\forall \mathbf{x} \in \Lambda, D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\Sigma},\mathbf{c}}(\Lambda)}$$

We also note $D_{\Lambda,\sigma,\mathbf{c}} = D_{\Lambda,\sigma\cdot I_n,\mathbf{c}}$. We omit the subscript $\mathbf{c}$ when the Gaussian is centered on $\mathbf{0}$ and the subscript Σ when $\Sigma = 1$. We will note D_1 the special and very important distribution that is the *continuous* spherical Gaussian over $\mathbb{R}^n$ centered on $\mathbf{0}$ and of standard deviation $\sigma = 1$. For $c \in \mathbb{R}, \sigma > 0$, we note $\lfloor c \rceil_\sigma$ the distribution $D_{\mathbb{Z},\sigma,c}$, and this notation is generalized coefficient-wise when c is replaced with a vector $\mathbf{c}$.

As an illustration, Figure 2.2 gives the continuous Gaussian D_1 in $\mathbb{R}$ and its discretization $D_{\mathbb{Z}} = D_{\mathbb{Z},1,0} = \lfloor 0 \rceil_1$ over $\mathbb{Z}$.

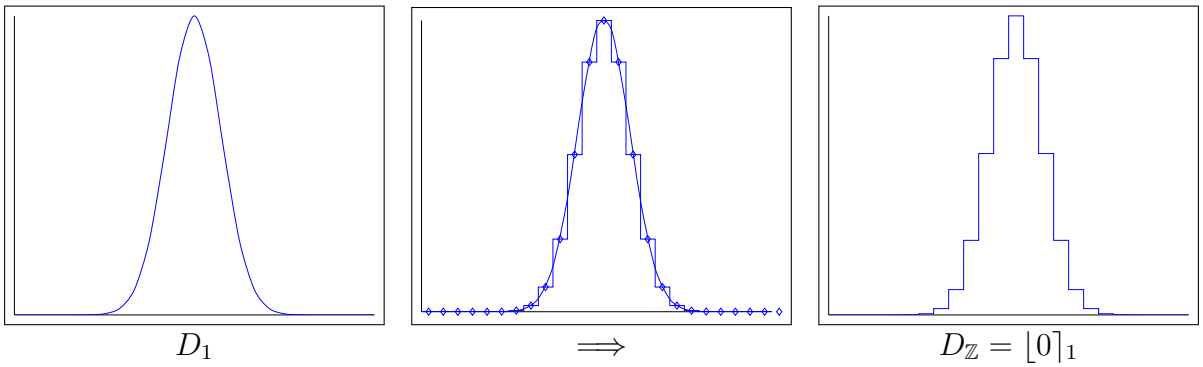


Figure 2.2: Continuous and Discrete Gaussians.

We also recall the definition of the smoothing parameter (and of a scaled version better suited to our purposes), as well as two lemmas which will be very useful through this thesis.

Definition 2.30 (Smoothing parameter [MR07])**.** *Let Λ be any n-dimensional lattice and $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho_{1/s\sqrt{2\pi},\mathbf{0}}(\Lambda^\star \setminus 0) \leqslant \epsilon$. We also define a scaled version $\eta'_\epsilon(\Lambda) \triangleq \frac{1}{\sqrt{2\pi}}\eta_\epsilon(\Lambda)$.*

Lemma 2.31 (Corollary of [MR07], Lemma 4.4)**.** *Let Λ be any n-dimensional lattice. Then for any $\epsilon \in (0,1), \sigma \geqslant \eta'_\epsilon(\Lambda)$ and $\mathbf{c} \in \mathbb{R}^n$, we have*

$$\rho_{\sigma,\mathbf{c}}(\Lambda) \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \cdot \rho_\sigma(\Lambda)$$

Lemma 2.32 (Special case of [MR07], Lemma 4.4)**.** *For any $\epsilon \in (0,1)$:*

$$\eta'_\epsilon(\mathbb{Z}^n) \leqslant \frac{1}{\pi}\sqrt{\frac{1}{2}\log\left(2n\left(1+\frac{1}{\epsilon}\right)\right)}$$

2.4 Gaussian Samplers

Now that we know what a discrete Gaussian distribution over a lattice is, we will review the current algorithms which allow to do that. Such algorithms are known as Gaussian samplers, and prior to this thesis there were two main Gaussian samplers.

The first one is Klein's sampler, which was introduced by Klein in [Kle00], and used in a cryptographic context for the first time by Gentry, Peikert and Vaikuntanathan [GPV08]. The second one is Peikert's sampler, introduced by the eponymous author in [Pei10]. They respectively generalize Babai's nearest plane and round-off algorithms and turn these *solvers*, who originally return one lattice point, into *samplers*, who return a point according to a Gaussian discretized over the lattice.

Both samplers inherit some properties of the algorithms they are based on. Klein's sampler is inherently sequential and Peikert's sampler is parallelizable, and in addition Peikert's sampler enjoys a quasilinear speed-up on several families of structured lattices. On the other hand Peikert's sampler outputs longer vectors, which lessens the security of cryptographic schemes based on it as it becomes easier for an attacker to imitate someone using this sampler.

2.4.1 Klein's Sampler

Klein's algorithm is a randomized version of Babai's NearestPlane algorithm. Though not stated in the paper in which it first appears [Kle00], it is explicitly acknowledged in [GPV08]. The only difference between the two is that the rounding step is replaced by a *randomized rounding* according to a discrete Gaussian over $\mathbb{Z}$: instead of performing $z_i \leftarrow \lfloor d_i \rceil_{\sigma_i}$, the new algorithms performs $z_i \leftarrow \lfloor d_i \rceil_{\sigma_i}$, where $\sigma_i = \sigma / \|\tilde{\mathbf{b}}_i\|$.

Algorithm 2.6 KleinSampler$(\mathbf{B}, \sigma, \mathbf{c})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathbb{Z}^{n \times m}$, its GSO $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$, a standard deviation σ, target $\mathbf{c} \in \mathbb{R}^m$

Ensure: $\mathbf{v}$ sampled in $D_{\Lambda(\mathbf{B}),\sigma,\mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$
$\mathbf{v}_n \leftarrow \mathbf{0}$
for $i \leftarrow n, ..., 1$ **do**
  $d_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$
  $\sigma_i \leftarrow \sigma / \|\tilde{\mathbf{b}}_i\|$
  $z_i \leftarrow \lfloor d_i \rceil_{\sigma_i}$
  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \mathbf{b}_i$
  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \mathbf{b}_i$
end for
return $\mathbf{v}_0$

In the new algorithm, for a fixed entry KleinSampler$(\mathbf{B}, \sigma, \mathbf{c})$ there is no longer one possible output. Instead the output distributions is statistically close to a discrete Gaussian. In Figure 2.3, we illustrate the transformation from NearestPlane to KleinSampler.

Figure 2.3 provides an informal explanation as to why $\sigma_i = \sigma / \|\tilde{\mathbf{b}}_i\|$: if one had chosen $\sigma_i = \sigma_0$ for some constant σ_0 instead (as done in the middle figure), then one can show that the output distribution would not be a spherical Gaussian but would instead leak the geometry of the orthogonalized basis $\tilde{\mathbf{B}}$. Using this imperfect sampler in a cryptographic

context would expose it to statistical attacks *à la* [NR06, Wan10, DN12b], though it would arguably be more difficult to recover the basis $\mathbf{B}$. So σ_i is scaled in order to get a spherical discrete Gaussian.

A more formal statement of KleinSampler's correctness is given in Theorem 2.33.

Theorem 2.33 (Theorem 1 of [DN12a], Concrete version of [GPV08, Th. 4.1])**.** *Let $m \geqslant n, \lambda$ be positive integers and $\epsilon = 2^{-\lambda}$. For any basis $\mathbf{B} \in \mathbb{Z}^{n \times m}$ and any target vector $\mathbf{c} \in \mathbb{R}^m$, the statistical distance between the output distribution of $\mathsf{KleinSampler}(\mathbf{B}, \sigma, \mathbf{c})$ and the perfect discrete Gaussian $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$ is upper bounded by $2^{-\lambda}$, provided:*

$$\sigma \geqslant \eta'_\epsilon(\mathbb{Z}^n) \cdot |\tilde{\mathbf{B}}| \quad \text{where} \quad \eta'_\epsilon(\mathbb{Z}^n) \approx \frac{1}{\pi} \cdot \sqrt{\frac{1}{2} \log\left(2n(1 + \frac{1}{\epsilon})\right)}.$$

We carefully note that Theorem 2.33 only guarantees that $\mathsf{KleinSampler}(\mathbf{B}, \sigma, \mathbf{c})$ behaves like a true Gaussian if σ is bigger than a fixed bound. This becomes false when σ becomes small. Indeed, one can check that if that was the case, then $\mathsf{KleinSampler}(\mathbf{B}, \sigma, \mathbf{c})$ would behave like a CVP-solver when σ tends to 0, which would be very surprising since CVP is NP-hard to approximate to a constant factor.

Instead, one can show – it is rather immediate from the description of both algorithms – that $\mathsf{KleinSampler}(\mathbf{B}, 0, \mathbf{c}) = \mathsf{NearestPlane}(\mathbf{B}, \mathbf{c})$ which does not solve CVP, even in dimension 2.

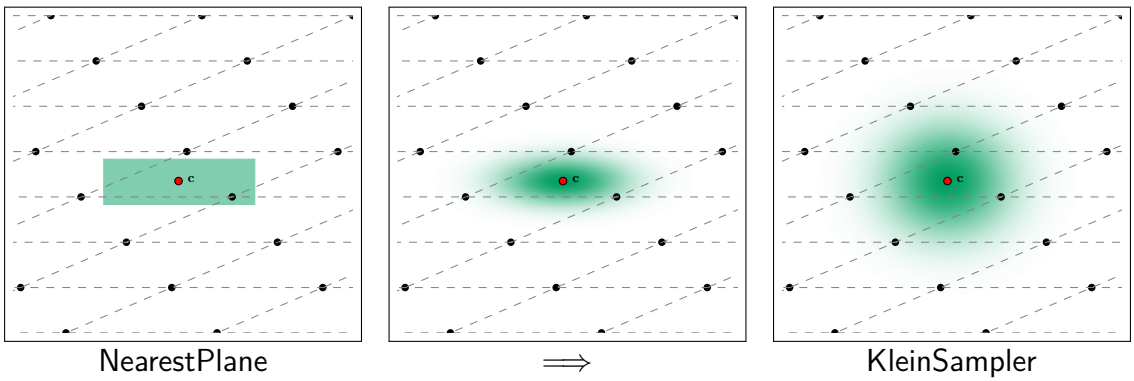


Figure 2.3: From NearestPlane to Klein's sampler.

2.4.2 Peikert's Sampler

Just like Klein's sampler randomizes the nearest plane algorithm, Peikert's sampler randomizes the round-off algorithm.

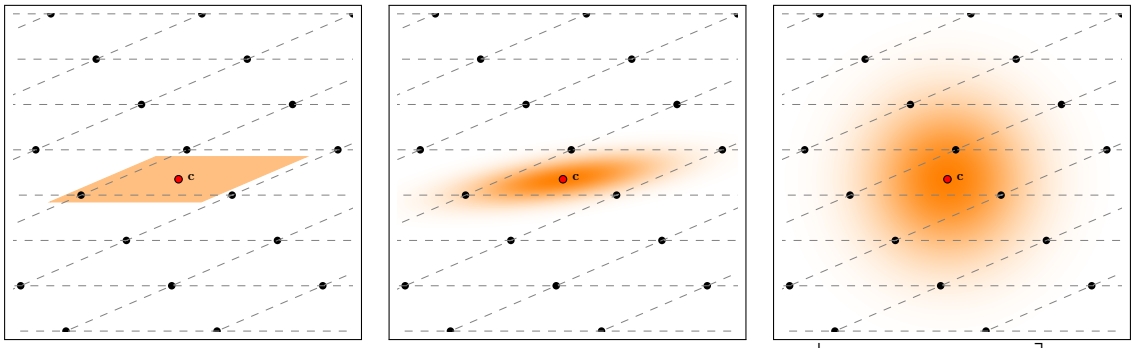


Figure 2.4: From Babai's RoundOff algorithm (on the left) to Peikert's sampler (on the right).

Algorithm 2.7 PeikertSampler$(\mathbf{B}, \sqrt{\Sigma}, \mathbf{c})$

Require: Basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$ of a lattice Λ, a rounding parameter r, a covariance matrix $\Sigma > \Sigma_1 = r^2 \mathbf{B}\mathbf{B}^t$, a matrix $\mathbf{C} = \sqrt{\Sigma_2} \in \mathbb{Z}^{n \times n}$ where $\Sigma_2 = \Sigma - \Sigma_1 > \mathbf{0}$, a target $\mathbf{c} \in \mathbb{R}^n$

Ensure: $\mathbf{z}$ sampled in $D_{\mathcal{L}(\mathbf{B}), \sqrt{\Sigma}, \mathbf{c}}$

1: $\mathbf{y} \leftarrow D_1 \cdot \mathbf{C}$ // Offline phase

2: **return** $\mathbf{z} \leftarrow \lfloor (\mathbf{c} - \mathbf{y})\mathbf{B}^{-1} \rceil_r \mathbf{B}$ // Online phase

Peikert's sampler outputs longer vectors than Klein's but is faster on several families of structured lattices. We do not elaborate here on this fact which is folklore and developed in the ulterior Section 5.3.2. In a nutshell it follows from the fact that Peikert's sampler only performs matrix-vector multiplications, and that when the matrices are structured, these multiplications can be done in time $O(n \log n)$ instead of $O(n^2)$.

In [Pei10], Peikert propose another slightly different algorithm, which he qualifies as optimized for q-ary lattices (which are integral lattices mod q).

Algorithm 2.8 PeikertSampler_q$(\mathbf{B}, \sqrt{\Sigma}, \mathbf{c})$

Require: Basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$ of a q-ary lattice Λ, a rounding parameter r, a covariance matrix $\Sigma > \Sigma_1 = r^2(4\mathbf{B}\mathbf{B}^t + \mathbf{I})$, a matrix $\mathbf{C} = \sqrt{\Sigma_2}$ where $\Sigma_2 = \Sigma - \Sigma_1 > \mathbf{0}$, a matrix $\mathbf{Z} = q \cdot \mathbf{B}^{-1} \in \mathbb{Z}^{n \times n}$ a target $\mathbf{c} \in \mathbb{Z}^n$

Ensure: $\mathbf{z}$ sampled in $D_{\mathcal{L}(\mathbf{B}), \sqrt{\Sigma}, \mathbf{c}}$

1: $\mathbf{y} \leftarrow \lfloor D_1 \cdot \mathbf{C} \rceil_r$ // Offline phase

2: **return** $\mathbf{z} \leftarrow \left\lfloor \frac{(\mathbf{c}-\mathbf{y})\mathbf{Z}}{q} \right\rceil_r \mathbf{B}$ // Online phase

The correctness of PeikertSampler (Alg. 2.7) is given by the following theorem.

Theorem 2.34 (Theorem 3.1 of [Pei10], continuous case, adapted)**.** *Let $\Sigma_1, \Sigma_2 > \mathbf{0}$ be positive definite matrices, with $\Sigma = \Sigma_1 + \Sigma_2 > \mathbf{0}$. Let Λ_1 be a lattice such that $\sqrt{\Sigma_1} \geqslant \eta'_\epsilon(\Lambda_1)$ for some positive $\epsilon \leqslant 1/2$, and let $\mathbf{c}$ be arbitrary. Consider the following probabilistic experiment:*

$$\textit{Choose } \mathbf{y} \sim D_{\sqrt{\Sigma_2}}, \textit{ then choose } \mathbf{z} \sim D_{\Lambda_1, \sqrt{\Sigma_1}, \mathbf{c}-\mathbf{y}}.$$

The statistical distance between the distribution of $\mathbf{z}$ and $D_{\Lambda_1, \sqrt{\Sigma}, \mathbf{c}}$ is upper bounded by 8ϵ.

Similarly to Theorem 2.33 for Klein's sampler, Theorem 2.34 means that the statistical distance between $\mathsf{PeikertSampler}(\mathbf{B}, \sqrt{\Sigma}, \mathbf{c})$ and the perfect Gaussian $D_{\mathcal{L}(\mathbf{B}),\sqrt{\Sigma},\mathbf{c}}$ is upper bounded by $2^{-\lambda}$ provided that resp. $\sqrt{\Sigma} > \eta'_\epsilon(\mathbb{Z}^n) \cdot s_1(\mathbf{B})$, for $\epsilon = 2^{-\lambda}$.

By following the definition of discrete Gaussians, one can check that for $\mathbf{c} \in \mathbb{Z}_n, \mathbf{c} = \mathbf{c}_0\mathbf{B}$, one gets the same distribution by doing either of the following:

1. Outputs $\mathbf{x} \leftarrow D_{\mathcal{L}(\mathbf{B}),\sqrt{\Sigma},\mathbf{c}}$
2. Samples $\mathbf{x}_0 \leftarrow D_{\mathbb{Z}^n,\sqrt{\mathbf{B}^{-1}\Sigma\mathbf{B}^{-t}},\mathbf{c}\mathbf{B}^{-1}}$ and outputs $\mathbf{x} \leftarrow \mathbf{x}_0\mathbf{B}$

A Note on Peikert's Samplers Both Peikert's samplers are much simpler when $\mathbf{B} = \mathbf{I}$, it is therefore often more convenient to use them over $\mathbb{Z}^n$ and to multiply the result by $\mathbf{B}$, since one spares a matrix-vector product in the process.

Moreover, proceeding like this brings out the difference between both samplers, which in our opinion is much more subtle than stated in [Pei10]. Both samplers are online/offline algorithms[5]. However, as illustrated by Figure 2.5, PeikertSampler_q performs a randomized rounding in the offline phase, so that in the online phase the center of each randomized rounding may only be in $\frac{1}{q}\mathbb{Z}$ (instead of $\mathbb{R}$).

	Generic basis $\mathbf{B}$	Special case $\mathbf{B} = \mathbf{I}$
PeikertSampler :	$\lfloor(\mathbf{c} - D_1 \cdot \mathbf{C})\mathbf{B}^{-1}\rceil_r\mathbf{B}$	$\lfloor\mathbf{c} - D_1 \cdot \mathbf{C}\rceil_r$
PeikertSampler_q :	$\lfloor(\mathbf{c} - \lfloor D_1 \cdot \mathbf{C}\rceil_r)\mathbf{B}^{-1}\rceil_r\mathbf{B}$	$\lfloor\mathbf{c} - \lfloor D_1 \cdot \mathbf{C}\rceil_r\rceil_r$

Figure 2.5: The output of Peikert's samplers. Here $D_1 = \mathcal{N}(0, 1)$ is the standard normal distribution. Offline computations are shown in *blue*.

In [Pei10], Peikert emphasizes that PeikertSampler_q (Alg. 2.8) is faster than PeikertSampler (Alg. 2.7) for q-ary lattices if the algorithms are used in an offline/online setting (this is clear on Figure 2.5). While we agree with this statement, this thesis doesn't consider offline/online properties of the samplers (despite the fact that one of our algorithms is faster in this setting too). In this context, PeikertSampler is faster and returns shorter vectors than PeikertSampler_q so we discard the latter and will only study the former in the rest of this thesis.

2.5 Gaussian Sampling in Cryptology

Now that we have Gaussian samplers – which are (probabilistic) polynomial algorithms for sampling discrete Gaussians – we will see some of their applications in cryptology. Most of these applications are in cryptographic constructions, but they are also used as black boxes in security reductions. They also have applications in cryptanalysis, with recent developments on this front. For the rest of the thesis, $\mathsf{Sample}(\mathbf{B}, \sigma, \mathbf{c})$ denotes an oracle for the distribution $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$. Such an oracle can be ideal or simulated with a Gaussian sampler.

[5]Online/offline digital signatures were introduced by Even et al. in [EGM90]. The idea is that a significant part of the computation necessary to sign a message (or answer a challenge) is done prior to the signer having knowledge of the message to sign. This property is useful when the signer is given queries at an irregular rate and/or has limited computational power: such situations occur per example for web servers and smart cards. A more detailed explanation can be found in [Yu08, Introduction].

2.5.1 Hash-and-sign Signature

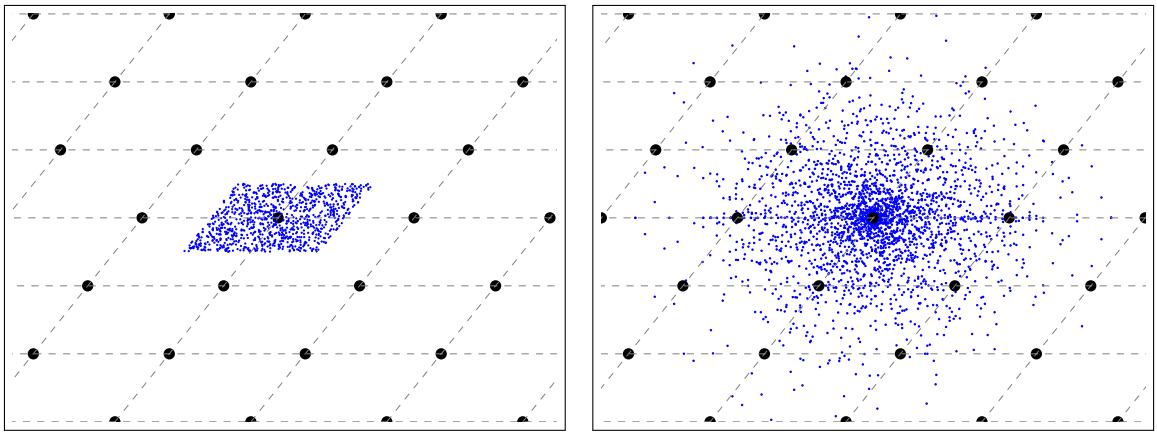
The first use of Gaussian sampling in cryptographic constructions is in [GPV08], where it was used for full-domain hash (FDH) signatures.

- $\mathsf{KeyGen}(1^n)$: let $(\mathbf{A}, \mathbf{B}) \leftarrow \mathsf{TrapGen}(1^n)$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ is a short basis of $\Lambda_q^{\perp}(\mathbf{A})$. The verification key is $\mathbf{A}$ and the signing key is $\mathbf{B}$.
- $\mathsf{Sign}(\mathbf{B}, \mathbf{m})$: if $(\mathbf{m}, \mathsf{Sign}(\mathbf{m}))$ is in local storage, output $\mathsf{Sign}(\mathbf{m})$. Otherwise:
  1. Find $\mathbf{c} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \times \mathbf{c}^t = H(\mathbf{m})$.
  2. Sample $\mathbf{v} \leftarrow \mathsf{Sample}(\mathbf{B}, \sigma, \mathbf{c})$.
  3. $\mathbf{s}(\mathbf{m}) \leftarrow \mathbf{c} - \mathbf{v}$. Store $(\mathbf{m}, \mathbf{s}(\mathbf{m}))$ and output $\mathbf{s}(\mathbf{m})$.
- $\mathsf{Verify}(\mathbf{A}, \mathbf{m}, \mathbf{s})$: if $\mathbf{A} \times \mathbf{s}^t = H(\mathbf{m})$ and $\|\mathbf{s}\| \leqslant \sigma\sqrt{2\pi m}$, accept. Else, reject.

Figure 2.6: A Stateful Full-Domain Hash Signature Scheme

The framework proposed in [GPV08] is impervious to the statistical attacks that broke NTRUSign and GGH (see Figure 2.7). Moreover, it was shown to be provably secure.

Theorem 2.35. *[GPV08, Propositions 6.1 and 6.2] The signature schemes described in Figures 2.8 and 2.6 are strongly existentially unforgeable under a chosen-message attack in the random oracle model if SIS is hard.*



GGH [GGH97], NTRUSign[HHGP+03] "GPV" framework [GPV08]

Figure 2.7: Distribution of the $(H(\mathbf{m}_i) - \mathsf{Sign}(\mathbf{m}_i))_i$

The fundamental reason given in [GPV08] for the stateful nature of the scheme above is that statefulness is needed to assert the security of their scheme under the SIS assumption.

While this reason alone is sufficient to make the scheme stateful, we show that the danger in removing the statefulness is not only theoretical, but also practical: being able to generate multiple valid signatures for the same hashed message poses a serious threat to the security of the scheme.

The strategy described below shows how allowing a user to query multiple signatures for a same (hashed) message would weaken the existential unforgeability of the scheme and leak information about the private key:

1. Query k signatures $\mathbf{s}_1, ..., \mathbf{s}_k$. With overwhelming probability, these signatures have a norm $\|\mathbf{s}\| \leqslant \sigma\sqrt{2\pi m}$.

2. Compute $\mathbf{v}_{i,j} = \mathbf{s}_i - \mathbf{s}_j$. Each $\mathbf{v}_{i,j}$ is a lattice point of $\Lambda_q^{\perp}(\mathbf{A}) = \Lambda_q(\mathbf{B})$ and with overwhelming probability, $\|\mathbf{v}_{i,j}\| \leqslant 2\sigma\sqrt{2\pi m}$.

3. Find m independent vectors $\mathbf{v}_{i,j}$'s to form a full-rank set $\mathbf{V}$, ans use Lemma 2.36 (Lemma 7.1 of [MG02]) to form a basis basis $\mathbf{B}'$ of $\Lambda_q(\mathbf{B})$.

4. Using Babai's nearest plane algorithm, this new basis $\mathbf{B}' = \{\mathbf{b}'_1, ..., \mathbf{b}'_m\}$ can be used to forge signatures $\mathbf{s}'(\mathbf{m})$ for any message $\mathbf{m}$, verifying the following bound:

$$\|\mathbf{s}'(m)\| \leqslant \frac{1}{2}\sqrt{\sum_i \|\tilde{\mathbf{b}}'_i\|^2} \leqslant \frac{1}{2}\sqrt{\sum_i \|\mathbf{b}'_i\|^2} \leqslant m\sigma\sqrt{2\pi}$$

We carefully note that the strategy described above does not explicitly break the security of the stateless variant of the scheme, as the signatures forged are $\sqrt{m}$ times bigger than required to be accepted.

However, this is a worst-case behavior. The strategy can be refined in many ways: per example, step 1 can be repeated to get signatures as small as possible, basis reduction algorithms such as LLL or its randomized variant BKZ [SE94] can be applied to the basis $\mathbf{B}'$ between steps 3 and 4, and at step 4 methods such as Kannan's embedding technique (see [Kan87], [LSL13] or [Gal12, Chapter 18]), may yield better results than the nearest plane algorithm.

2.5.1.1 Stateless Variant

For the reasons we just stated, a straightforward stateless version of the scheme described above may not be considered as secure. However, statefulness is often considered an undesirable property. Here, the need to store all the signatures and the associated hashes raises at least two issues:

1. A storage issue: memory-limited entities like embedded devices – such as a router – may not be able to store thousands or millions of signatures. An unexpected loss of those signatures – or the corresponding identities – would also be problematic.
2. A communication issue: in the case where a signing key is shared between two or more entities, any of them asked to sign a message would need to make sure none of the others has already provided a signature for it.

To avoid statefulness, [GPV08] proposed to derandomize the Sign procedure, as done by [Gol87] for the Goldwasser-Micali-Rivest signature scheme. In our case, the internal randomness used by the Sample subroutine is replaced by the output of a pseudorandom function evaluated over the concatenation of the message (or its hash) and an additional secret seed.

Another solution, also proposed in [GPV08], is to randomize the hash. It is less constaining than the previous solution. However, when turning the GPV signature scheme into an IBE scheme, this method no longer applies, as opposed to the previous one.

The idea is to add a salt $r \in \{0,1\}^k$ to the message before hashing it. In [GPV08], the authors propose to take $k = m$ and claim that $k = \omega(\log n)$ suffices for asymptotic security. However, to avoid birthday attacks, it is also necessary that $k \geqslant 2\lambda$, where λ is the security parameter. The solution is detailed in Figure 2.8.

- $\mathsf{KeyGen}(1^n)$: let $(\mathbf{A}, \mathbf{B}) \leftarrow \mathsf{TrapGen}(1^n)$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ is a short basis of $\Lambda_q^{\perp}(\mathbf{A})$. The verification key is $\mathbf{A}$ and the signing key is $\mathbf{B}$.
- $\mathsf{Sign}(\mathbf{B}, \mathbf{m})$:
  1. Choose $\mathbf{r} \leftarrow \{0,1\}^k$ and find $\mathbf{c} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\mathbf{c}^t = H(\mathbf{m} \| \mathbf{r})$.
  2. Sample $\mathbf{v} \leftarrow \mathsf{Sample}(\mathbf{B}, \sigma, \mathbf{c})$.
  3. $\mathbf{s}(\mathbf{m}) \leftarrow \mathbf{c} - \mathbf{v}$. Output $\mathbf{s}(\mathbf{m})$.
- $\mathsf{Verify}(\mathbf{A}, \mathbf{m}, \mathbf{s})$: if $\mathbf{A}\mathbf{s}^t = H(\mathbf{m})$ and $\|\mathbf{s}\| \leqslant \sigma\sqrt{2\pi n}$, accept. Else, reject.

Figure 2.8: A Probabilistic FDH Signature Scheme

Readers interested in the implementation of this signature scheme may read Chapter 6, where we give a detailed study of this scheme over NTRU lattices.

2.5.2 Identity-Based Encryption

In addition to their signature schemes, Gentry et al. also propose an identity-based encryption scheme in [GPV08], which can be can be viewed as an extension of their signature scheme. Indeed, a good Gaussian sampler – i.e. which samples close vectors – for $\Lambda_q^{\perp}(\mathbf{A})$ can be used to construct a partial trapdoor for any $\Lambda(\mathbf{A_h})$, where $\mathbf{A_h} \triangleq (\mathbf{A} \| \mathbf{h})$. More precisely, for any $\mathbf{h} \in \mathbb{Z}_q^n$ (in practice $\mathbf{h}$ will the hash of an identity), Gaussian sampling allows to sample a short vector $\mathbf{s}'$ of $\Lambda_q^{\perp}(\mathbf{A_h})$, as outlined in Figure 2.9.

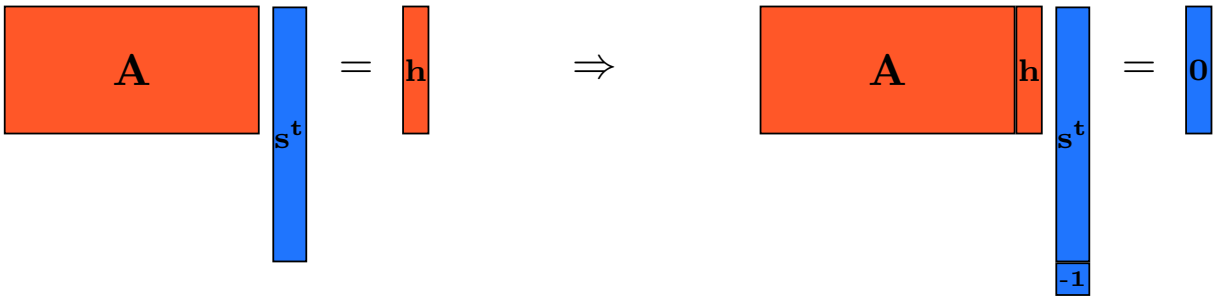


Figure 2.9: Constructing a partial trapdoor for $\mathbf{A_h} \triangleq (\mathbf{A} \| \mathbf{h})$

Gentry et al. proposed a variant of Regev's encryption scheme [Reg05] that uses $\mathbf{A_h}$ as an encryption key and $\mathbf{s}'$ as a decryption key, resulting in an identity-based encryption (IBE) scheme. As their encryption scheme switches the key generation and encryption parts of Regev's original scheme, they call it Dual Regev.

Figure 2.10 gives Gentry et al.'s IBE scheme as described in [GPV08]. Readers interested in this scheme may read Chapter 8, where we give further explanation of the mathematics in action, as well as a complete instantiation and implementation of it over NTRU lattices.

- $\mathsf{Setup}(1^n)$: let $(\mathbf{A}, \mathbf{B}) \leftarrow \mathsf{TrapGen}(1^n)$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ is a short basis of $\Lambda_q^{\perp}(\mathbf{A})$. The master public key is $\mathbf{A}$ and the master private key is $\mathbf{B}$.
- $\mathsf{Extract}(\mathbf{B}, \mathsf{id})$: if $(\mathsf{id}, \mathbf{sk}(\mathsf{id}))$ is in local storage, output $\mathbf{s}_{\mathsf{id}}$. Otherwise:
  1. Find $\mathbf{c} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\mathbf{c}^t = H(\mathsf{id})$.
  2. Sample $\mathbf{v} \leftarrow \mathsf{Sample}(\mathbf{B}, \sigma, \mathbf{c})$.
  3. $\mathbf{sk}(\mathsf{id}) \leftarrow \mathbf{c} - \mathbf{v}$. Store $(\mathsf{id}, \mathbf{sk}(\mathsf{id}))$ and output $\mathbf{sk}(\mathsf{id})$.
- $\mathsf{Encrypt}(\mathbf{A}, \mathsf{id}, b)$: To encrypt the bit b for the identity id:
  1. Let $\mathbf{h} = H(\mathsf{id}) \in \mathbb{Z}_q^n$.
  2. Choose $\mathbf{x} \leftarrow \mathbb{Z}_q^n$ uniformly, $\mathbf{e} \leftarrow \chi^m$ and $e \leftarrow \chi$.
  3. Set $\mathbf{p} = \mathbf{x} \cdot \mathbf{A} + \mathbf{e} \in \mathbb{Z}_q^m$ and $c = \mathbf{x} \cdot \mathbf{h}^t + \mathbf{e} + \lfloor \frac{q}{2} \rfloor b \in \mathbb{Z}_q$. Output $(\mathbf{p}, c)$.
- $\mathsf{Decrypt}(\mathbf{sk}, (\mathbf{p}, c))$: Compute $b' = c - \mathbf{p} \cdot \mathbf{sk}^t$. If b' is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ modulo q, output 0. Otherwise, output 1.

Figure 2.10: An Identity-Based Encryption Scheme

2.5.3 Basis Randomization and Applications to Standard Model Signatures, Hierarchical IBE and Attribute-Based Encryption

In 2010, Cash, Hofheinz, Kiltz and Peikert [CHKP10] used Gaussian sampling to create even more powerful cryptographic primitives, namely hierarchical identity-based encryption and standard model signatures. The underlying idea that allowed both primitives was *basis randomization* using Gaussian sampling.

Lemma 2.36 ([MG02], Lemma 7.1, page 129)**.** *There is a deterministic polynomial-time algorithm* $\mathsf{ToBasis}(\mathbf{S}, \mathbf{V})$ *that given a full-rank set* $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_m\}$ *of lattice vectors in* $\mathcal{L}(\mathbf{S})$*, outputs a basis* $\mathbf{T}$ *of* $\mathcal{L}(\mathbf{S})$ *such that* $\|\tilde{\mathbf{t}}_i\| \leqslant \|\tilde{\mathbf{v}}_i\|$ *for all* i*.*

Theorem 2.37. *Let* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *be a basis. There exists a polynomial-time algorithm* $\mathsf{RandBasis}(\mathbf{S}, \sigma)$ *which, given a basis* $\mathbf{S}$ *of the* m*-dimensional lattice* $\mathcal{L}_q^{\perp}(\mathbf{A})$ *and a parameter* $\sigma \geqslant |\tilde{\mathbf{S}}| \cdot \omega(\sqrt{\log n})$*, outputs a new basis* $\mathbf{S}'$ *of* $\mathcal{L}_q^{\perp}(\mathbf{A})$ *such that:*

- $|\tilde{\mathbf{S}}'| \leqslant \sigma \cdot \sqrt{m}$ *with probability* $1 - 2^{-\Omega(n)}$
- *The distribution of* $\mathbf{S}'$ *is independent of the input* $\mathbf{S}$*. More precisely, for any two valid inputs* $(\mathbf{S}_1, \sigma)$ *and* $(\mathbf{S}_2, \sigma)$*, the random variables* $\mathsf{RandBasis}(\mathbf{S}_1, \sigma)$ *and* $\mathsf{RandBasis}(\mathbf{S}_2, \sigma)$ *are statistically close.*

The algorithm $\mathsf{RandBasis}$ can be briefly described as follows:

1. $i \leftarrow 0$. While $i < m$:
   - $\mathbf{v} \leftarrow \mathsf{Sample}(\mathbf{S}, \sigma, \mathbf{0})$
   - If $\mathbf{v}$ is independent of $\{\mathbf{v}_1, ..., \mathbf{v}_i\}$, set $\mathbf{v}_i \leftarrow \mathbf{v}$ and $i \leftarrow i + 1$.

2. Use Lemma 2.36 to convert $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_m\}$ into a basis: $\mathbf{S}' \leftarrow \mathsf{ToBasis}(\mathbf{S}, \mathbf{V})$.

3. Output $\mathbf{S}'$.

In addition, [CHKP10] provides an algorithm that extends a short basis of a lattice into a short basis of another lattice closely related to the first one.

Lemma 2.38 ([CHKP10], Lemma 3.2)**.** *There is a deterministic polynomial-time algorithm* ExtBasis *such that, given:*

- *A matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *whose columns generate the entire group* $\mathbb{Z}_q^n$
- *A basis* $\mathbf{B} \in \mathbb{Z}_q^{m \times m}$ *of* $\mathcal{L}_q^\perp(\mathbf{A})$
- *A matrix* $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$

$\mathsf{ExtBasis}(\mathbf{B}, \mathbf{A}' = [\mathbf{A} \| \bar{\mathbf{A}}])$ *outputs a basis* $\mathbf{B}'$ *of* $\mathcal{L}_q^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$ *such that* $|\tilde{\mathbf{B}}'| = |\tilde{\mathbf{B}}|$.

Given a (efficiently computable and non-necessarily short) solution $\mathbf{W} \in \mathbb{Z}^{\bar{m} \times m}$ to the equation $\mathbf{A}\mathbf{W}^t = -\bar{\mathbf{A}}$, the matrix $\mathbf{B}'$ of Lemma 2.38 is very simple to construct:

$$\mathbf{B}' = \left[\begin{array}{c|c} \mathbf{B} & \mathbf{0} \\ \hline -\mathbf{W} & \mathbf{I} \end{array}\right]$$

Lemmas 2.37 and 2.38 allow to build a short basis of $\mathcal{L}_q^\perp(\mathbf{A}')$ given a short basis $\mathbf{B}$ of $\mathcal{L}_q^\perp(\mathbf{A})$, for any "extension" $\mathbf{A}' = [\mathbf{A} \| \bar{\mathbf{A}}]$ of $\mathbf{A}$ and, very importantly, in a way that does not reveal any information $\mathbf{B}$: first use Lemma 2.38 to extend $\mathbf{B}$ into a matrix $\mathbf{B}'$, then apply Gaussian sampling *via* Lemma 2.37 to randomize this basis.

We will not elaborate further but building on this technique – which makes a critical use of Gaussian sampling – unlocks sophisticated cryptographic constructions over lattices such as signatures in the standard model [CHKP10, Boy10b], hierarchical IBE [CHKP10, ABB10a, ABB10b] and attribute-based encryption [Boy13, BGG+14].

2.5.4 Hardness of Lattice Problems

In addition to the aforementioned cryptographic constructions, Gaussian sampling – and a naturally associated problem called DGS – has been used in many works as an intermediate tool to assess the hardness of lattice-related problems. We first give the definition of the Discrete Gaussian Problem:

Definition 2.39. *For* $\gamma \geqslant 1$ *and* $\epsilon \geqslant 0$*, a distribution* $\mathcal{P}$ *is said to be* (γ, ϵ)*-close to a distribution* $\mathcal{Q}$ *is there is another distribution* $\mathcal{P}'$ *with the same support as* $\mathcal{Q}$ *such that*

1. *The statistical distance between* $\mathcal{P}$ *and* $\mathcal{P}'$ *is at most* ϵ
2. *For any* $\mathbf{x}$ *in the support of* $\mathcal{Q}$*,* $\mathcal{Q}(\mathbf{x})/\gamma \leqslant \mathcal{P}(\mathbf{x}) \leqslant \gamma \cdot \mathcal{Q}(\mathbf{x})$

Definition 2.40 (DGS – Discrete Gaussian Problem)**.** *Given a lattice basis* $\mathbf{B}$*, a standard deviation* $\sigma > 0$ *and a vector* $\mathbf{t} \in \mathrm{Span}(\mathbf{B})$*, output a lattice vector* $\mathbf{v}$ *according to a distribution* (γ, ϵ)*-close to* $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{t}}$*.*

If $\mathbf{t} = \mathbf{0}$*, then this is a special instance called centered DGS problem.*

Since this thesis focuses on reducing the time and space complexities of Gaussian sampling algorithms which are already known to run in probabilistic polynomial-time, it has little to no impact on the works presented in this section, which assume the existence of an algorithm sampling a discrete Gaussian in polynomial time for some parameters – such an algorithm is known to exist at least since [GPV08] – and use it as a black box. However, these works help to put Gaussian sampling in a more generic context and to grasp a better understanding of the problems it solves, we therefore list a few of them:

- Klein's sampler was first used by himself in [Kle00] to solve BDD_α with a factor $\alpha = O(1/n)$ in polynomial time (under the condition that the lattice is already "pre-processed"). Building on Klein's work, Liu, Lyubashevsky and Micciancio [LLM06] improved his result to $\alpha = O(\sqrt{(\log n)/n})$.

- Regev uses DGS – which he qualifies as a "non-standard" problem – in [Reg05, Reg09] to show the quantum hardness of LWE, provided the hardness of standard *worst-case* lattice problems. Brakerski, Langlois, Stehlé, Peikert and Regev use it again in [BLP+13] to show the hardness of LWE, this time under *classical* reductions.

- Micciancio and Peikert [MP13] use Gaussian sampling as a subroutine to prove the hardness of SIS with a small modulus.

- A recent series of works by Aggarwal, Dadush, Regev and Stephens-Davidowitz use Gaussian sampling to solve in time $2^{n+o(n)}$ the SVP [ADRS15] and CVP [ADS15] problems. Stephens-Davidowitz showed [Ste15] that there is a dimension preserving equivalence between the DGS and approximate-CVP problems (resp. the centered DGS and approximate-SVP problem).

Part I

Improving Existing Gaussian Samplers

In this part of the thesis, we aim to improve the two existing Gaussian samplers and their understanding. A statistical analysis is performed in Chapter 3, and a geometrical approach is adopted in Chapter 4.

Chapter 3 performs a statistical analysis of the Gaussian samplers. Instead of the statistical distance, we rely on the KL divergence, which is a metric more suited to Gaussian distributions. This change of metric leads to two improvements. The first one is that we can sample with a standard deviation smaller by a factor $\sqrt{2}$ than previously thought, which in ulterior parts of this thesis will allow to gain between approximately 10 and 20 bits of security. The second one is that we can lower the required floating-point precision and even avoid completely the use of high-precision arithmetic for some cryptographic parameters – assuming we can process in hardware floating-point numbers with 64-bits significands, which is the case in many recent families of processors. This not only makes the samplers faster but also easier to implement.

Practical lattice-based constructions always use specific families of lattices that are structured over rings. Chapter 4 analyzes the geometric structure of these lattices. It turns out that we can use an algorithm known as the Levinson recursion [Lev47] to perform the Gram-Schmidt orthogonalization $O(n)$ times faster over these lattices. Moreover, reverting the execution order of the Levinson recursion allows to divide the storage requirement of Klein's sampler by a factor $O(n) \approx 300$ for cryptographic parameters, from about $100Mb$ to less than $1Mb$.

Chapter 3 develops ideas that were part of a joint paper with Léo Ducas and Vadim Lyubashevsky, ***Efficient Identity-Based Encryption over NTRU Lattices*** [DLP14], which was published at ASIACRYPT. Chapter 4 covers almost the totality of a joint paper with Vadim Lyubashevsky, ***Quadratic Time, Linear Space Algorithms for Gram-Schmidt Orthogonalization and Gaussian Sampling in Structured Lattices*** [DLP14], which was published at EUROCRYPT.

Chapter 3

Improved Parameters by using the Kullback-Leibler Divergence

3.1 Introduction

The statistical distance is an omnipresent tool in cryptography. One of its key properties is that for any distributions $\mathcal{P}, \mathcal{Q}$ over a common support X and any measurable event E, we have the following inequality:

$$\mathcal{Q}(E) \geqslant \mathcal{P}(E) - \Delta(\mathcal{P}, \mathcal{Q}) \tag{3.1}$$

A very useful abstraction is to modelize a cryptographic scheme as relying on some distribution $\mathcal{P}$ and the success of an attacker against this scheme as an event E: if $\Delta(\mathcal{P}, \mathcal{Q})$ is negligible, the same scheme will be equally hard to break if $\mathcal{P}$ is replaced by $\mathcal{Q}$. Combined with its other properties – described in Section 2.3.2 –, it makes the statistical distance a powerful tool for security proofs.

However, other metrics exist that may be used in a similar way to make security arguments. The one this chapter will focus on is the Kullback-Leibler divergence (or KL divergence). It shares several properties with the statistical distance, like preservation under any transformation and sub-additivity, as well as a weaker form of the equation 3.1. This makes the KL divergence a viable candidate to replace the statistical distance in many security arguments.

The intuition behind using KL-divergence can be described by the following example. If $\mathcal{B}_c$ denotes a Bernoulli variable with probability c on 1, then trying to distinguish with constant probability $\mathcal{B}_{1/2+\epsilon/2}$ from $\mathcal{B}_{1/2}$ requires $O(1/\epsilon^2)$ samples. Therefore if there is no adversary who can forge in time less than t (for some $t > 1/\epsilon^2$) on a signature scheme where some parameter comes from the distribution $\mathcal{B}_{1/2}$, then we can conclude that no adversary can forge in time less than approximately $1/\epsilon^2$ if that same variable were distributed according to $\mathcal{B}_{1/2+\epsilon/2}$. This is because a successful forger is also clearly a distinguisher between the two distributions (since forgeries can be checked), but no distinguisher can work in time less than $1/\epsilon^2$. On the other hand, distinguishing $\mathcal{B}_\epsilon$ from $\mathcal{B}_0$ requires only $O(1/\epsilon)$ samples. And so if there is a time t forger against a scheme using $\mathcal{B}_0$, all one can say about a forger against the scheme using $\mathcal{B}_\epsilon$ is that he cannot succeed in time less than $1/\epsilon$. In both cases, however, we have statistical distance ϵ between the two distributions. In this regard, statistical distance based arguments are not tight; but the KL-divergence is finer grained and can give tighter proofs. Indeed, in the first case, we can set $1/\epsilon$ to be

the square root of our security parameter, whereas in the second case, $1/\epsilon$ would have to be the security parameter.

More concretely, just like the statistical distance, one can argue that a $2^{-\lambda}$-secure scheme relying on a distribution $\mathcal{P}$ will be remain about $2^{-\lambda}$-secure if $\mathcal{P}$ is replaced by $\mathcal{Q}$, provided that their KL divergence is less than $2^{-\lambda}$.

The KL divergence is not fundamentally a "better tool" than the statistical distance for evaluating the security of cryptographic constructions. There exists distributions for which it is smaller than the statistical distance (and hence provides tighter security arguments), but sometimes it is the other way around, as illustrated by Figure 3.1.

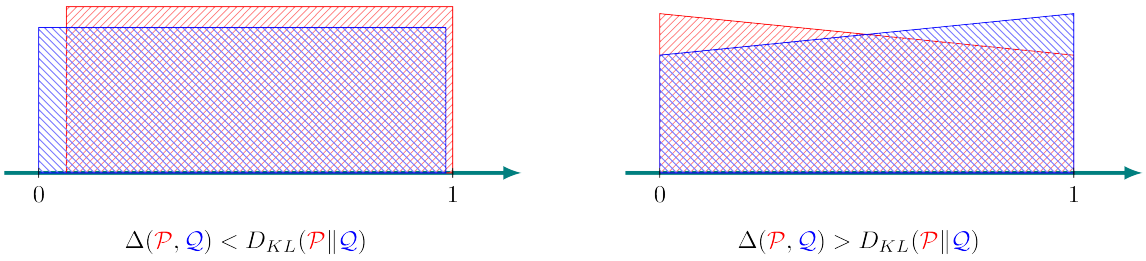


Figure 3.1: Distributions for which the statistical distance (resp. KL divergence) is smaller.

When studying discrete and continuous Gaussians, the KL divergence turns out to be much smaller than the statistical distance in many cases. For Gaussian samplers, favoring a KL divergence-based analysis over a statistical distance-based analysis allows to to:

- Sample with a standard deviation smaller by a factor $\sqrt{2}$ than a statistical distance-based analysis (Section 3.3).
- Work with a floating-point precision smaller by a factor 2 (Section 3.4).

However, it is important to notice that the security arguments we will give – which can be summarized with "a $2^{-\lambda}$-secure scheme using $\mathcal{P}$ remains so using $\mathcal{Q}$ if the KL divergence between $\mathcal{P}$ and $\mathcal{Q}$ is less than $2^{-\lambda}$)" – hold only when breaking the scheme is a search problem. More details are given in Section 3.2.1.

3.1.1 Roadmap

First, Section 3.2 recalls the definition and some properties of the KL divergence and give security arguments based on it. Then in Section 3.3, we show how the KL divergence allows to use Gaussian samplers with a smaller standard deviation. We evaluate in Section 3.4 the *maximal* precision necessary to run Gaussian samplers. Section 3.5 concludes this chapter and gives some leads that could possibly improve the results of this work or of other works.

3.1.2 Related Works

Ducas and Nguyen [DN12a] considered using lazy evaluation in Gaussian samplers to ensure they use high-precision floating-point arithmetic only with a small probability, and rely on low-precision arithmetic most of the time.

The use of alternative measures to the statistical distance has been widespread in cryptography since several years, with a good introduction being [Cac97]. In the context

of lattice-based cryptography, the Kullback-Leibler divergence has recently been used [PDG14] by Pöppelmann, Ducas and Güneysu to lower the storage requirements given in [DDLL13].

A number of papers have also used Rényi divergence R_a, a generalized divergence which coincides with the Kullback-Leibler divergence for $a = 1$. Among those works, Lyubashevsky, Peikert and Regev [LPR13a] made use of the Rényi divergence in the search to decision reduction of the RLWE problem [LPR10]. Langlois, Stehlé and Steinfeld [LSS14, Lan14] employed it to reduce the parameters of the candidate multilinear map of Garg, Gentry and Halevi [GGH13]. Ling, Phan, Stehlé and Steinfeld [LPSS14] used it to analyze the $k-$ LWE problem, a variant of the LWE problem.

More recently, Bai, Langlois, Lepoint, Stehlé and Steinfeld [BLL+15] proposed several applications of the Rényi divergence in lattice-based cryptography, which include reducing the parameters for the signature and encryption schemes of [GPV08], the BLISS signature scheme [DDLL13, PDG14] and an alternative and tighter proof for the hardness of the LWE problem with noise uniform (in an interval) [DMQ13].

3.2 Preliminaries: The Kullback-Leibler Divergence

In this section, we present the notion of Kullback-Leibler divergence (or KL divergence) and explain how it can replace the statistical distance. We also state its limitations as a security metric, and how to overcome them.

Definition 3.1 (Kullback-Leibler Divergence)**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions over a common countable set Ω, and let $S \subset \Omega$ be the strict support of $\mathcal{P}$ ($\mathcal{P}(i) > 0$ iff $i \in S$). The Kullback-Leibler divergence, noted D_{KL} of $\mathcal{Q}$ from $\mathcal{P}$ is defined as:*

$$D_{KL}(\mathcal{P}\|\mathcal{Q}) = \sum_{i \in S} \log\left(\frac{\mathcal{P}(i)}{\mathcal{Q}(i)}\right)\mathcal{P}(i)$$

with the convention that $\log(x/0) = +\infty$ for any $x > 0$.

For complements the reader can refer to [CT91, Tsy08]. Just like the statistical distance, the KL divergence is positive definite, sub-additive and preserved under any injective transformation:

- $D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{Q}) \geqslant 0$, with an equality iff $\mathcal{P} = \mathcal{Q}$
- $D_{\mathrm{KL}}(\mathcal{P}_0 \times \mathcal{P}_1\|\mathcal{Q}_0 \times \mathcal{Q}_1) = D_{\mathrm{KL}}(\mathcal{P}_0\|\mathcal{Q}_0) + D_{\mathrm{KL}}(\mathcal{P}_1\|\mathcal{Q}_1)$
- For any function f, the following inequality holds and is an equality when f is injective over the support of $\mathcal{P}$:
$$D_{\mathrm{KL}}(f(\mathcal{P})\|f(\mathcal{Q})) \leqslant D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{Q})$$

A useful lemma linking the KL divergence to the statistical distance is Pinsker's inequality, for which a proof can be found in [Tsy08, Chapter 2, Lemma 2.5].

Theorem 3.2 (Pinsker's Inequality)**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions on a countable set. Then*

$$\Delta(\mathcal{P}, \mathcal{Q}) \leqslant \sqrt{\frac{1}{2} D_{KL}(\mathcal{P}\|\mathcal{Q})}$$

3.2.1 Security Argument from the Kullback-Leibler Divergence

Considering the example of the introduction, Pinsker's inequality allows to get the bound

$$\mathcal{Q}(E) \geqslant \mathcal{P}(E) - \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\mathcal{P} \| \mathcal{Q})} \tag{3.2}$$

which is very similar to the one involving the statistical distance. To conclude a security argument using KL divergence, we however rely on a slightly different formulation, which is easier to use "in a black box".

Corollary 3.3 (Bounding Success Probability Variations)**.** *Let $\mathcal{E}^{\mathcal{P}}$ be an algorithm making at most q queries to an oracle distribution $\mathcal{P}$, and $\mathcal{D}_{\mathcal{P}}$ the output distribution of $\mathcal{E}^{\mathcal{P}}$. Let $\epsilon \geqslant 0$, $\mathcal{Q}$ be a distribution such that $D_{KL}(\mathcal{P} \| \mathcal{Q}) < \epsilon$. Then*

$$\Delta(\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{Q}}) \leqslant \sqrt{\frac{q\epsilon}{2}}$$

Proof. We have:

$$\Delta(\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{Q}}) \leqslant \Delta(\mathcal{P}^q, \mathcal{Q}^q) \leqslant \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\mathcal{P}^q \| \mathcal{Q}^q)} \leqslant \sqrt{\frac{q}{2} D_{\mathrm{KL}}(\mathcal{P} \| \mathcal{Q})} \leqslant \sqrt{\frac{q\epsilon}{2}}$$

The first inequality comes from the preservation of the statistical distance under any function (Prop. 2.29), the second one from Pinsker's inequality (Thm. 3.2) and the third one from the sub-additivity of the KL divergence. □

Security Argument. This corollary lets us conclude that if a scheme is λ-bit secure with access to a perfect oracle for distribution $\mathcal{P}$, then it is also about λ-bit secure with oracle access to $\mathcal{Q}$ if $D_{\mathrm{KL}}(\mathcal{P} \| \mathcal{Q}) \leqslant 2^{-\lambda}$.

Indeed, consider a search problem $\mathcal{S}^{\mathcal{Q}}$ using oracle access to a distribution $\mathcal{Q}$, and assume it is not λ-bit hard; that is there exists an attacker $\mathcal{A}$ that solve $\mathcal{S}^{\mathcal{Q}}$ with probability p and has running time less than $2^{\lambda}p$; this implies (by repeating the attack until success) an algorithm $\mathcal{A}'$ that solves $\mathcal{S}^{\mathcal{Q}}$ in time $\approx 2^{\lambda}$ with probability at least $3/4$.

Such algorithms make $q \leqslant 2^{\lambda}$ queries to $\mathcal{Q}$. If $D_{\mathrm{KL}}(\mathcal{P} \| \mathcal{Q}) \leqslant 2^{-\lambda}$, Corollary 3.3 ensures that the success of $\mathcal{A}'$ against $\mathcal{S}^{\mathcal{P}}$ will be at least $1/4$; in other word if $\mathcal{S}^{\mathcal{Q}}$ is λ-bit secure, $\mathcal{S}^{\mathcal{P}}$ is also about λ-bit secure.

Remark 3.4. The security argument we give hold only for search problems, not decisional ones. One can of course use Pinsker's inequality to get a bound on the statistical distance, which then gives security arguments for decision problems. However, in our usecases the KL divergence's value will typically be about the square of the statistical distance (so it will be twice smaller on a logarithmic scale). Using Pinsker's inequality directly would cancel this square factor, rendering pointless the use of the KL divergence. Therefore the KL divergence has to be handled with caution when used for arguing the security of a cryptographic scheme: to have efficient proofs, one has to make sure that the problem underlying the security of the scheme is a search problem.

3.2.2 Dealing with the Asymmetry and Lack of Triangle Inequality

The KL divergence is not symmetric, neither does it verify the triangle inequality. Yet that does not prevent us in any way to use it to use it in security arguments. Indeed,

both Pinsker's inequality and Corollary 3.3 turn a bound over the KL divergence of two distributions into a bound over their statistical distance, which is symmetric and verifies the triangle inequality.

As an example, let $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ be three distributions such that $(D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{Q}) \leqslant 2^{-\lambda}$ or $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) \leqslant 2^{-\lambda})$ and $(D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{R}) \leqslant 2^{-\lambda}$ or $D_{\mathrm{KL}}(\mathcal{R}\|\mathcal{Q}) \leqslant 2^{-\lambda})$. Let $\mathcal{E}^{\mathcal{P}}$ be an algorithm making at most q queries to $\mathcal{P}$ and having an output distribution $\mathcal{D}_{\mathcal{P}}$. Applying Corollary 3.3 twice yields $\Delta(\mathcal{D}_{\mathcal{P}}, \mathcal{D}_{\mathcal{R}}) \leqslant \sqrt{2q2^{-\lambda}}$, and the security argument following Corollary 3.3 allows to argue that a cryptographic scheme $2^{-\lambda}$-secure with either $\mathcal{P}, \mathcal{Q}$ or $\mathcal{R}$ will stay about $2^{-\lambda}$-secure with any of the other distributions.

3.3 Reducing the Standard Deviation of Gaussian Samplers

In Section 2.4, we recalled two theorems (Theorems 2.33 and 2.34) that can informally be interpreted as follows: if a cryptographic scheme relying on a discrete Gaussian oracle $D_{\mathcal{L}(\mathbf{B}),\sigma,\cdot}$ (resp. $D_{\mathcal{L}(\mathbf{B}),\sqrt{\Sigma},\cdot}$) is $2^{-\lambda}$-secure, then it remains so if we use $\mathsf{KleinSampler}(\mathbf{B}, \sigma, \cdot)$ (resp. $\mathsf{PeikertSampler}(\mathbf{B}, \sqrt{\Sigma}, \cdot)$) as an oracle for the discrete Gaussian, provided that $\sigma > \eta'_\epsilon(\mathbb{Z}^n) \cdot |\tilde{\mathbf{B}}|$ (resp. $\sqrt{\Sigma} > \eta'_\epsilon(\mathbb{Z}^n) \cdot s_1(\mathbf{B})$), for $\epsilon = 2^{-\lambda}$.

This imposes a lower bound on the standard deviation that can be used with each sampler. In this section, we show that in both cases, we can take σ smaller by a factor of roughly $\sqrt{2}$. The statistical distance will no longer be upper bounded by $2^{-\lambda}$, but the KL divergence will still be, so we can still sample with a smaller standard deviation and have security arguments. Mechanically, this increases the security of cryptographic schemes relying on these samplers.

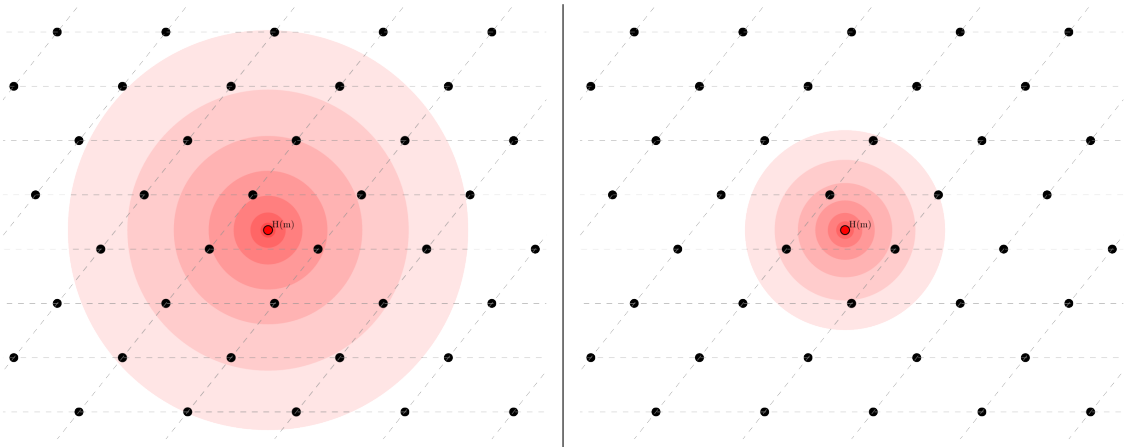


Figure 3.2: Using the KL divergence in a security analysis allows to sample smaller Gaussians.

To study the KL divergence between the samplers and a perfect Gaussian, the following lemma will be useful.

Lemma 3.5 (KL divergence for bounded ratio [PDG14])**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions of same countable support. Assume that for any $i \in S$, there exists some $\delta(i) \in (0, 1/4)$*

such that we have a bounded ratio $\mathcal{Q}(i)/\mathcal{P}(i)$*:* $|\mathcal{P}(i) - \mathcal{Q}(i)| \leqslant \delta(i)\mathcal{P}(i)$*. Then*

$$D_{KL}(\mathcal{P}\|\mathcal{Q}) \leqslant 2\sum_{i\in S}\delta(i)^2\mathcal{P}(i).$$

For distributions with a bounded ratio $\delta < 1/4$, Lemma 3.5 allows to get a KL divergence of about $2\delta^2$, which is much smaller

Section 3.3.1 show that we can take a standard deviation smaller by a factor $\sqrt{2}$ for Klein's sampler, and Section 3.3.2 does the same for Peikert's sampler.

3.3.1 Klein's Sampler

This section proves that we can use Klein's Sampler with a standard deviation smaller (by a factor of $\sqrt{2}$) than claimed in [DN12a].

Theorem 3.6 (KL Divergence of Klein's Sampler)**.** *For any* $\epsilon \in (0, 1/4n)$*, if* $\sigma \geqslant \eta'_\epsilon(\mathbb{Z})\cdot|\tilde{\mathbf{B}}|$ *then the KL divergence between* $D_{\Lambda(\mathbf{B}),\mathbf{c},\sigma}$ *and the output of* $\mathsf{KleinSampler}(\mathbf{B},\sigma,\mathbf{c})$ *is bounded by* $2\left(1-\left(\frac{1+\epsilon}{1-\epsilon}\right)^n\right)^2 \approx 8n^2\epsilon^2$*.*

Proof. The probability that Klein's algorithm outputs $\mathbf{x} = \tilde{\mathbf{x}}$ on input $(\sigma, \mathbf{B}, \mathbf{c})$ is proportional to

$$\prod_{i=1}^{n}\frac{1}{\rho_{\sigma_i,c'_i}(\mathbb{Z})}\cdot\rho_{\sigma,\mathbf{c}}(\tilde{\mathbf{x}})$$

for $\sigma_i = \sigma/\|\tilde{\mathbf{b}}_i\|$ and some $c'_i \in \mathbb{R}$ that depends on $\mathbf{c}$ and $\mathbf{B}$. as detailed in [GPV08]. By assumption, $\sigma_i \geqslant \eta_\epsilon(\mathbb{Z})$, therefore $\rho_{\sigma_i,c'_i}(\mathbb{Z}) \in [\frac{1-\epsilon}{1+\epsilon}, 1]\cdot\rho_{\sigma_i}(\mathbb{Z})$ (see [MR04, Lemma 4.4]). The relative error to the desired distribution (proportional to $\rho_{\sigma,\mathbf{c}}(\tilde{\mathbf{x}})$) is therefore bounded by $1-\left(\frac{1+\epsilon}{1-\epsilon}\right)^n$; we can conclude using Lemma 2 from [PDG14]. □

As a direct consequence, we can use Klein's sampler with a standard deviation $\sigma \approx \eta'_\epsilon(\mathbb{Z}^n)|\tilde{\mathbf{B}}|$ with $\epsilon = 2^{-\lambda/2}$ instead of $\epsilon = 2^{-\lambda}$, which results in dividing σ by roughly $\sqrt{2}$.

3.3.2 Peikert's Sampler

This section proves that we can use Peikert's Sampler with a standard deviation smaller (by a factor of $\sqrt{2}$) than claimed in [Pei10].

Theorem 3.7. *Reprising the notations of Theorem 2.34, the statistical distance (resp. KL divergence) between the distribution of* $\mathbf{z}$ *and* $D_{\Lambda_1,\sqrt{\Sigma},\mathbf{c}}$ *is upper bounded by* $\frac{\epsilon}{1-\epsilon} \approx \epsilon$ *(resp.* $2\left(\frac{2\epsilon}{1-\epsilon}\right)^2 \approx 8\epsilon^2$*).*

Proof. We only improve a specific point of the original theorem: namely, adding a bound on the KL divergence and very slightly enhancing the bound on the statistical distance. We invite the reader interested in the complete proof to read [Pei10].

Let $\bar{\mathbf{z}} \in \Lambda_1$. From the proof of theorem 3.1 of [Pei10], we have $Pr[\bar{\mathbf{z}} = \mathbf{z}] \propto \rho_{\sqrt{\Sigma}}(\bar{\mathbf{z}})\cdot\left[1, \frac{1+\epsilon}{1-\epsilon}\right]$ (and not $\left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right]$ like in the original proof, since in the continuous case we need only in invoke Lemma 2.4 of [Pei10] once). It follows that $Pr[\bar{\mathbf{z}} = \mathbf{z}] \in \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right]\cdot D_{\Lambda_1,\sqrt{\Sigma},\mathbf{c}}(\bar{\mathbf{z}})$. A straightforward computation (resp. Lemma 3.5) yields the bound on the statistical distance (resp. the KL divergence). □

Just as for Klein's sampler, we can divide σ by $\sqrt{2}$, as long as Peikert's sampler is used in a search problem.

3.4 Precision Analysis of Gaussian Samplers

In this section, we perform a KL divergence-based analysis of the precision required to securely use Klein's and Peikert's samplers. Compared with a statistical distance-based analysis (which can be trivially obtained using Pinsker's inequality), this allows to get a precision twice lower.

3.4.1 Ratio of Gaussians Sums in $\mathbb{Z}$

In this section, we give a lemma which will be useful to evaluate the precision necessary for Klein's and Peikert's samplers. First, we need the two following lemmas, first stated by Micciancio and Regev in [MR07]. A generalization of the first one can be found in [Lyu08].

Lemma 3.8. *[MR07, Lemma 4.2] Let Λ be a n-dimensional lattice, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^n$ a vector of norm 1 and reals $\epsilon \in (0,1)$, $\sigma \geqslant 2\eta'_\epsilon(\Lambda)$. The following inequalities hold:*

$$\left|\mathbb{E}_{\mathbf{x} \leftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle]\right| \leqslant \frac{\sqrt{2\pi}\epsilon\sigma}{1-\epsilon}$$

$$\left|\mathbb{E}_{\mathbf{x} \leftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle^2] - \sigma^2\right| \leqslant \frac{2\pi\epsilon\sigma^2}{1-\epsilon}$$

Lemma 3.9. *[MR07, Lemma 4.4] Let Λ be a n-dimensional lattice, $\mathbf{c} \in \mathbb{R}^n$, and reals $\epsilon \in (0,1)$, $\sigma \geqslant \eta'_\epsilon(\Lambda)$. We have:*

$$\mathbb{P}_{\mathbf{x} \leftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\|\mathbf{x}-\mathbf{c}\| \geqslant \sigma\sqrt{2\pi n}] \leqslant \frac{1+\epsilon}{1-\epsilon}2^{-n}$$

The following lemma bounds the ratio of two Gaussian sums in $\mathbb{Z}$ with slightly different centers and standard deviations.

Lemma 3.10. *Let $c, \bar{c} \in \mathbb{R}$, $\sigma, \bar{\sigma} > 0$. Let $\rho(z) \triangleq \rho_{\sigma,c}(z)$, $\bar{\rho}(z) \triangleq \rho_{\bar{\sigma},\bar{c}}(z)$, $\mathcal{D}(z) \triangleq \rho(z)/\rho(\mathbb{Z})$ and $\bar{\mathcal{D}}(z) \triangleq \bar{\rho}(z)/\bar{\rho}(\mathbb{Z})$. Let $u(z) = \frac{(z-\bar{c})^2}{2\bar{\sigma}^2} - \frac{(z-c)^2}{2\sigma^2}$. Then*

$$e^{-\mathbb{E}_{z\leftarrow\mathcal{D}}[u]} \leqslant \frac{\bar{\rho}(\mathbb{Z})}{\rho(\mathbb{Z})} \leqslant e^{-\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}}[u]}$$

Proof. We first prove the left inequality. We have

$$\begin{aligned} \bar{\rho}(z) &= e^{-u(z)}\rho(z) \\ \Rightarrow \frac{\bar{\rho}(z)}{\rho(\mathbb{Z})} &= e^{-u(z)}\mathcal{D}(z) \\ \Rightarrow \frac{\bar{\rho}(\mathbb{Z})}{\rho(\mathbb{Z})} &= \mathbb{E}_{z\leftarrow\mathcal{D}}[e^{-u(z)}] \\ \Rightarrow \frac{\bar{\rho}(\mathbb{Z})}{\rho(\mathbb{Z})} &\geqslant e^{-\mathbb{E}_{z\leftarrow\mathcal{D}}[u(z)]} \end{aligned} \tag{3.3}$$

Where the last inequality comes from Jensen's inequality: since e is convex, $\mathbb{E}[e^{-u}] \geqslant e^{\mathbb{E}[-u]}$. Following the same reasoning, one gets

$$\left(\bar{\mathcal{D}}(z)e^{u(z)} = \frac{\rho(z)}{\bar{\rho}(\mathbb{Z})}\right) \Rightarrow \left(\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}}[e^u] = \frac{\rho(\mathbb{Z})}{\bar{\rho}(\mathbb{Z})}\right) \Rightarrow \left(\frac{\bar{\rho}(\mathbb{Z})}{\rho(\mathbb{Z})} \leqslant e^{-\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}}[u]}\right)$$

□

3.4.2 Analysis of Klein's Sampler

In this section, we evaluate the precision necessary to run Klein's sampler in finite precision and get an output distribution indistinguishable from a sampler with infinite precision.

We recall that the algorithm that we analyze, KleinSampler, is the Algorithm 2.6 of Section 2.4.1. First, Lemma 3.11 bounds the value of the d_i's computed at step 4 of the algorithm.

Lemma 3.11. *Let $\delta \in [0, .01)$. Let $\mathbf{t} \in \mathbb{R}^m$, $\mathbf{b}, \bar{\mathbf{b}} \in \mathbb{R}^m$ and the precomputed values $\|\mathbf{b}\|, \|\bar{\mathbf{b}}\| \in \mathbb{R}^+$ be such that:*

- $\|\mathbf{b} - \bar{\mathbf{b}}\| \leqslant \delta \|\mathbf{b}\|$
- $|\|\mathbf{b}\| - \|\bar{\mathbf{b}}\|| \leqslant \delta \|\mathbf{b}\|$

In addition, let $c = \frac{\langle \mathbf{t}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2}$ (resp. $\bar{c} = \frac{\langle \mathbf{t}, \bar{\mathbf{b}} \rangle}{\|\bar{\mathbf{b}}\|^2}$). then

$$|c - \bar{c}| \leqslant 3.3 \frac{\|\mathbf{t}\|}{\|\mathbf{b}\|} \delta$$

Proof.

$$\bar{c} = \left(\underbrace{\frac{\langle \mathbf{t}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2}}_{c} + \underbrace{\frac{\langle \mathbf{t}, \bar{\mathbf{b}} - \mathbf{b} \rangle}{\|\mathbf{b}\|^2}}_{|\cdot| \leqslant \frac{\|\mathbf{t}\| \|\mathbf{b}\|}{\|\mathbf{b}\|^2} \delta} \right) \cdot \underbrace{\frac{\|\bar{\mathbf{b}}\|^2}{\|\mathbf{b}\|^2}}_{\in [(1-\delta)^2, (1+\delta)^2]}$$

Since $|c| \leqslant \frac{\|\mathbf{t}\|}{\|\mathbf{b}\|}$, we have $|\bar{c} - c| \leqslant |(|c| + \frac{\|\mathbf{t}\|}{\|\mathbf{b}\|}\delta) \cdot (1+\delta)^2 - |c|| \leqslant 3.3 \frac{\|\mathbf{t}\|}{\|\mathbf{b}\|}\delta$. □

The following lemma bounds the KL divergence between two instantiations of Klein's sampler over the same vector $\mathbf{c}$: one instantiation using perfect precomputed values, and one with imperfect values. Without loss of generality, we can admit that the output $\mathbf{v}$ of the sampler is such that $\|\mathbf{v} - \mathbf{c}\|_2 \leqslant \sigma\sqrt{2\pi m}$. For the parameters in the range of Lemma 3.12, Lemma 3.9 guarantees that this happens with overwhelming probability.

Lemma 3.12. *Let $q, m, n \in \mathbb{N}^\star$ such that $m \geqslant n \geqslant 128$, $\mathbf{B} \in \mathbb{Z}^{n \times m}, \mathbf{c} \in \mathbb{Z}_q^m$, $\epsilon \in (0, .01)$ and $\sigma \geqslant \eta'_\epsilon(\mathbb{Z}) \cdot |\tilde{\mathbf{B}}|$. Let $\mathcal{E}$ (resp. $\bar{\mathcal{E}}$) be* KleinSampler$(\mathbf{B}, \sigma, \mathbf{c})$ *executed with perfect precomputed values $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_i)_i, (\sigma_i)_i$ (resp. imperfect precomputed values $\bar{\tilde{\mathbf{B}}} = (\bar{\tilde{\mathbf{b}}}_i)_i, (\bar{\sigma}_i)_i$).*[1] *Let $\delta \in [0, .01)$ be such that:*

1. $\frac{3.3q\sqrt{m}}{\min_i \|\tilde{\mathbf{b}}_i\|} \delta \leqslant 0.01$
2. $\forall i \in [\![1, n]\!]$, $\|\tilde{\mathbf{b}}_i - \bar{\tilde{\mathbf{b}}}_i\| \leqslant \|\tilde{\mathbf{b}}_i\| \cdot \delta$
3. $\forall i \in [\![1, n]\!]$, $|\sigma_i - \bar{\sigma}_i| \leqslant |\sigma_i| \cdot \delta$

Let $\mathcal{D}$ (resp. $\bar{\mathcal{D}}$) denote the output distribution of $\mathcal{E}$ (resp. $\bar{\mathcal{E}}$). Let

$$C = \frac{12nq^2m\delta^2}{\sigma^2} + \frac{10\delta n}{\sigma}\left[q\sqrt{m} + \sigma\right] \cdot \left(1 + \frac{\epsilon}{1-\epsilon}\right)$$

The KL divergence between $\mathcal{D}$ and $\bar{\mathcal{D}}$ is bounded as follows:

$$D_{KL}(\bar{\mathcal{D}} \| \mathcal{D}) \leqslant (e^C - 1)^2$$

[1] For each tuple, product or sum indexed by i, we implicitly consider i to span $[\![1, n]\!]$.

Remark 3.13. In practice, the expression of the bound for the KL divergence can often be simplified: if $\delta, \epsilon \ll 1$, then

$$e^C - 1 \approx C \approx \frac{10\delta n}{\sigma}\left[q\sqrt{m} + \sigma\right]$$

Proof. Let $\mathbf{v} = \sum_i \hat{z}_i \mathbf{b}_i \in \mathcal{L}(\mathbf{B})$ be a possible output of both samplers.To sample $\mathbf{v}$, there exist a unique n-tuple $(c_i)_i$ (resp. $(\bar{c}_i)_i$) such that at step i, $\mathcal{E}$ (resp. $\bar{\mathcal{E}}$) samples a discrete Gaussian in $\mathbb{Z}$ around c_i (resp. $\bar{c}_i$). The probability that $\mathbf{v}$ is output by $\mathcal{E}$ is $\mathcal{D}(\mathbf{v}) = \prod_i \mathcal{D}_i(\hat{z}_i) = \prod_i \frac{\rho_i(\hat{z}_i)}{\rho_i(\mathbb{Z})}$, where $\rho_i \overset{\Delta}{=} \rho_{\mathbb{Z},\sigma_i,c_i}$ is uniquely defined by $\mathbf{v}$. Similarly, $\bar{\mathcal{D}}(\mathbf{v}) = \prod_i \frac{\bar{\rho}_i(\hat{z}_i)}{\bar{\rho}_i(\mathbb{Z})}$, where $\bar{\rho}_i \overset{\Delta}{=} \rho_{\mathbb{Z},\bar{\sigma}_i,\bar{c}_i}$.

The subject of this proof is to upper and lower bound $\frac{\mathcal{D}(\mathbf{v})}{\bar{\mathcal{D}}(\mathbf{v})}$, which then allows to use Lemma 3.5. We have

$$\frac{\mathcal{D}(\mathbf{v})}{\bar{\mathcal{D}}(\mathbf{v})} = \prod_i \frac{\rho_i(\hat{z}_i)}{\rho_i(\mathbb{Z})}\frac{\bar{\rho}_i(\mathbb{Z})}{\bar{\rho}_i(\hat{z}_i)} = \prod_i \frac{\rho_i(\hat{z}_i)}{\bar{\rho}_i(\hat{z}_i)}\frac{\bar{\rho}_i(\mathbb{Z})}{\rho_i(\mathbb{Z})}$$

For each i, let $u_i(z) \overset{\Delta}{=} \frac{(z-\bar{c}_i)^2}{2\bar{\sigma}_i^2} - \frac{(z-c_i)^2}{2\sigma_i^2}$. Lemma 3.10 yields:

$$e^{-\mathbb{E}_{z\leftarrow\mathcal{D}_i}[u_i]} \leqslant \frac{\bar{\rho}_i(\mathbb{Z})}{\rho_i(\mathbb{Z})} \leqslant e^{-\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[u_i]}$$

So that we have:

$$\sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z\leftarrow\mathcal{D}_i}[u_i]\right] \leqslant \log\left(\frac{\mathcal{D}(\mathbf{v})}{\bar{\mathcal{D}}(\mathbf{v})}\right) \leqslant \sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[u_i]\right] \tag{3.4}$$

We now bound the left part of equation 3.4. Let $A = \sum_i[u_i(\hat{z}_i) - \mathbb{E}_{z\leftarrow\mathcal{D}_i}[u_i]]$. We write $\bar{\sigma}_i = (1+\delta_{\sigma_i})\sigma_i$, where each $|\delta_{\sigma_i}| \leqslant \delta$ by hypothesis. Developing u_i yields:

$$u_i(z_i) = \frac{1}{2(1+\delta_{\sigma_i})^2\sigma_i^2}\left[(c_i-\bar{c}_i)^2 + 2(c_i-\bar{c}_i)(z_i-c_i) - (2\delta_{\sigma_i}+\delta_{\sigma_i}^2)(z_i-c_i)^2\right] \tag{3.5}$$

Replacing each u_i in A by the expression of equation 3.5 yields:

$$\begin{aligned}
A &= \textstyle\sum_i \frac{1}{2(1+\delta_{\sigma_i})^2\sigma_i^2} &&\left[2(c_i-\bar{c}_i)(\hat{z}_i-c_i-\mathbb{E}_{z_i\leftarrow\mathcal{D}_i}[z_i-c_i]) \right. &&\left. -(2\delta_{\sigma_i}+\delta_{\sigma_i}^2)[(\hat{z}_i-c_i)^2-\mathbb{E}_{z_i\leftarrow\mathcal{D}_i}[(z_i-c_i)^2]\right] \\
|A| &\leqslant \textstyle\sum_i \frac{1}{2(1+\delta_{\sigma_i})^2\sigma_i^2} &&\left[2|c_i-\bar{c}_i|(|\hat{z}_i-c_i|+\frac{\sqrt{2\pi}\epsilon\sigma_i}{1-\epsilon}) \right. &&\left. +(2\delta_{\sigma_i}+\delta_{\sigma_i}^2)[(\hat{z}_i-c_i)^2+\sigma_i^2+\frac{2\pi\epsilon\sigma_i^2}{1-\epsilon}]\right] \\
|A| &\leqslant \textstyle\sum_i \frac{1}{2(1+\delta_{\sigma_i})^2\sigma^2} &&\left[6.6q\sqrt{m}\delta(\|\tilde{\mathbf{b}}_i\|\cdot|\hat{z}_i-c_i|+\frac{\sqrt{2\pi}\epsilon\sigma}{1-\epsilon}) \right. &&\left. +(2\delta_{\sigma_i}+\delta_{\sigma_i}^2)[\|\tilde{\mathbf{b}}_i\|^2(\hat{z}_i-c_i)^2+\sigma^2+\frac{2\pi\epsilon\sigma^2}{1-\epsilon}]\right] \\
|A| &\leqslant \textstyle\frac{1}{2(1-\delta)^2\sigma^2} &&\left[6.6q\sqrt{m}\delta(\|\mathbf{v}-\mathbf{c}\|_1+\frac{\sqrt{2\pi}\epsilon n\sigma}{1-\epsilon}) \right. &&\left. +2.2\delta[\|\mathbf{v}-\mathbf{c}\|_2^2+n\sigma^2+\frac{2\pi\epsilon n\sigma^2}{1-\epsilon}]\right] \\
|A| &\leqslant \textstyle\frac{(1.1)^2}{\sigma} &&\left[3q\sqrt{m}\delta(n\sqrt{2\pi}+\frac{\sqrt{2\pi}\epsilon n}{1-\epsilon}) \right. &&\left. +\delta[2n\pi\sigma+n\sigma+\frac{2\pi\epsilon n\sigma}{1-\epsilon}]\right]
\end{aligned} \tag{3.6}$$

In equation 3.6, the first line simply develops the formula for A. For the second line, we use Lemma 3.8 to bound the expected values. The third line uses the fact that $|c_i - \bar{c}_i| \leqslant 3.3\frac{\|\mathbf{c}_i\|}{\|\tilde{\mathbf{b}}_i\|}\delta \leqslant \frac{3.3q\sqrt{m}}{\|\tilde{\mathbf{b}}_i\|}\delta$. For the fourth line, we bound each $\frac{1}{(1+\delta_{\sigma_i})^2}$ with $\frac{1}{(1-\delta)^2}$ and notice that $\sum_i \|\tilde{\mathbf{b}}_i\|\cdot|\hat{z}_i - c_i| = \|\mathbf{v}-\mathbf{c}\|_1$ and $\sum_i \|\tilde{\mathbf{b}}_i\|^2\cdot(\hat{z}_i-c_i)^2 = \|\mathbf{v}-\mathbf{c}\|_2^2$ (both equalities follow directly from Lemma 4.4 of [GPV08]).

In the fifth line, we use the bounds $\|\mathbf{v}-\mathbf{c}\|_2 \leqslant \sigma\sqrt{2\pi n}$, and $\|\mathbf{v}-\mathbf{c}\|_1 \leqslant \sigma n\sqrt{2\pi}$: the first one comes from Lemma 3.9, and the second one follows from the fact that there exists

a vector $\mathbf{u}$ with coefficients being only ± 1 such that $\|\mathbf{v}-\mathbf{c}\|_1 = |\langle \mathbf{v}-\mathbf{c}, \mathbf{u}\rangle|$. Applying the Cauchy-Schwartz theorem yields the bound.

We now bound the second part of equation 3.4. We can also write u_i as follows:

$$u_i(z_i) = \frac{1}{\bar{\sigma}_i^2}\left[-(1+\delta_{\sigma_i})^2(c_i-\bar{c}_i)^2 + 2(1+\delta_{\sigma_i})^2(c_i-\bar{c}_i)(z_i-c_i) - (2\delta_{\sigma_i}+\delta_{\sigma_i}^2)(z_i-\bar{c}_i)^2\right] \tag{3.7}$$

Now, let $B = \sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[u_i]\right]$ be the right term of equation 3.4. To bound B, we replace the u_i in each $u_i(\hat{z}_i)$ by the expression in equation 3.6, and the u_i in each $\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[u_i]$ by the expression of equation 3.7. This yields:

$$\begin{aligned} |B| \leqslant & \sum_i \frac{1.1(c_i-\bar{c}_i)^2}{\bar{\sigma}_i^2} \\ & + \sum_i \frac{1.1}{2\sigma_i^2}\left[2|c_i-\bar{c}_i|\cdot|\hat{z}_i-c_i| + |2\delta_{\sigma_i}+\delta_{\sigma_i}^2|\cdot|\hat{z}_i-c_i|^2\right] \\ & + \sum_i \frac{1}{2\sigma_i^2}\left[2|c_i-\bar{c}_i|\cdot|\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[z_i-\bar{c}_i]| + |2\delta_{\sigma_i}+\delta_{\sigma_i}^2|\cdot\mathbb{E}_{z\leftarrow\bar{\mathcal{D}}_i}[(z_i-\bar{c}_i)^2]\right] \\ \leqslant & \frac{12nq^2m\delta^2}{\sigma^2} + \frac{(1.1)^2}{\sigma}\left[3q\delta n\sqrt{2\pi m} + 2\delta n\pi\sigma\right] \\ & + \frac{(1.1)^2}{\sigma}\left[3q\sqrt{m}\delta\left(\tfrac{\sqrt{2\pi}\epsilon n}{1-\epsilon}\right) + \delta\left[n\sigma + \tfrac{2\pi\epsilon n\sigma}{1-\epsilon}\right]\right] \end{aligned} \tag{3.8}$$

The bounds hereinabove are derived exactly as for $|A|$. This gives the bound $|A|, |B| \leqslant C$, which in turns allows to bound $\frac{\mathcal{D}}{\bar{\mathcal{D}}}$: $e^{-C} \leqslant \frac{\mathcal{D}}{\bar{\mathcal{D}}} \leqslant e^{C}$. We can conclude using lemma 3.5. □

Interpretation. If the inputs of Klein's sampler are precise up to a given precision δ, then the sampler with finite precision will be indistinguishable from the same sampler with infinite precision. By the triangle inequality, this implies indistinguishability from a perfect Gaussian if the standard deviation σ is taken large enough. Concrete numbers can be found in Table 3.1.

Table 3.1: Precision for Klein's sampler.

This table gives upper bounds on the precision sufficient for an imprecise Klein's sampler to be indistinguishable (*i.e.* $D_{\mathrm{KL}} \leqslant 2^{-\lambda}$) from a perfect Klein's sampler. The precisions are computed for $NTRU$ lattices of dimension $2N = 2048$, modulus $q = 1024$.

Security Level λ	Precision $\lvert\log_2 \delta\rvert$ for $\tilde{\mathbf{B}}$
80	65
128	89
192	121

3.4.3 Analysis of Peikert's Sampler

In this section, we analyze the precision requirements of Peikert's sampler when sampling from a spherical Gaussian. For the reasons explained before, we don't distinguish the offline and online phases, and therefore privilege the first version – namely PeikertSampler – of the sampler (Algorithm 2.7 of Section 2.4.2).

Let us consider sampling from a discrete Gaussian: $\mathbf{z} \leftarrow D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$. As explained in Section 2.4.2, Algorithm 2.7 is equivalent to these three steps:

$\mathsf{Peikert}(\mathbf{B}, \sqrt{\Sigma}, \mathbf{c})$:
1. $\mathbf{x} \leftarrow D_1^n$
2. $\mathbf{y} \leftarrow \mathbf{x} \cdot \mathbf{C}$, where $\mathbf{C} = \sqrt{\Sigma - r^2\mathbf{B}\mathbf{B}^t}\mathbf{B}^{-1}$
3. $\mathbf{z} \leftarrow \lfloor \mathbf{c}\mathbf{B}^{-1} - \mathbf{y} \rceil_r \mathbf{B}$

Figure 3.3: Algorithm equivalent to PeikertSampler

Since in practice these operations are performed in floating-point arithmetic (FPA), we prefer this description: less operations are involved, and therefore the analysis is much simpler.

The only values that might induce FPA imprecisions are $\mathbf{x}$ and $\mathbf{C}$.[2] Indeed, $\mathbf{x}$ is sampled from a *continuous* Gaussian. As for $\mathbf{C}$, it is a square root matrix which has to be computed by methods using either real or very large rational numbers.

To model the fact that in practice, Peikert's sampler uses an oracle for D_1 that truncates the result to a finite number of bits of precision, we use the following definition.

Definition 3.14. *We recall that D_1 is the continuous centered Gaussian of standard deviation 1: $D_1 \sim \mathcal{N}(0,1)$. For any $0 \leqslant \delta < 1/2$, we note $\mathcal{N}_\delta$ any distribution sampling $x \leftarrow D_1$, and outputting some $\tilde{x} \in [1-\delta, 1+\delta] \cdot x$.*

Remark 3.15. We can assume without loss of generality that $\mathcal{N}_\delta^n$ samples a vector $\mathbf{x}$ of norm lesser than $\sqrt{2n}$. Indeed, each coefficient of $\mathbf{x}$ is independently sampled from $\mathcal{N}(0,1)$ so $\|\mathbf{x}\|^2 \sim \chi_n^2$, where χ_n^2 is the Chi-squared distribution with n degrees of freedom. Using tail bounds, we determine that $\|\mathbf{x}\| \geqslant \sqrt{kn}$ with probability at most $(ke^{1-k})^{n/2}$, so this sampler will be indistinguishable from the classical PeikertSampler as long as $(2e^{-1})^{n/2} < 2^{-\lambda}$.

$\mathcal{N}_0^n$ (resp. $\mathcal{N}_\delta^n$) will be used to modelize oracles giving a sample from D_1^n with infinitely many (resp. $\lfloor |\log_2 \delta| \rfloor$) bits of precision. Of course such $\mathcal{N}_0^n$ and $\mathcal{N}_\delta^n$ can be very easy to distinguish, but we will show that instantiations of PeikertSampler using one or the other are not.

The following lemma bounds the KL divergence between two instances of PeikertSampler with slightly different inputs:

Lemma 3.16. *Let $n \geqslant 128$, $\mathbf{x} = (x_i)_i \in \mathbb{R}^n$ such that $\|\mathbf{x}\| \leqslant \sqrt{2n}$, $\mathbf{C} \in \mathbb{R}^{n \times n}$ and $r \geqslant \eta'_\epsilon(\mathbb{Z}^n)$ for some $\epsilon \in (0,1)$. Let $\delta \in [0, .01)$, $\bar{\mathbf{x}} = (\bar{x}_i)_i \in \mathbb{R}^n$ and $\bar{\mathbf{C}} \in \mathbb{R}^{n \times n}$ be such that:*

- *For any $i \llbracket 1, n \rrbracket$, $|x_i - \bar{x}_i| \leqslant \delta \cdot |x_i|$*
- *$s_1(\mathbf{C} - \bar{\mathbf{C}}) \leqslant \delta \cdot s_1(\mathbf{C})$*

In addition, let $\mathcal{D}$ (resp. $\bar{\mathcal{D}}$) be the output distribution of $\lfloor \mathbf{x} \cdot \mathbf{C} \rceil_r$ (resp. $\lfloor \bar{\mathbf{x}} \cdot \bar{\mathbf{C}} \rceil_r$). Let:

$$C = \frac{8ns_1(\mathbf{C})\delta}{r} \cdot \left[1 + \frac{\epsilon}{1-\epsilon} + \frac{2s_1(\mathbf{C})\delta}{r}\right]$$

The KL divergence between $\mathcal{D}$ and $\bar{\mathcal{D}}$ is bounded as follows:

$$D_{KL}(\bar{\mathcal{D}} \| \mathcal{D}) \leqslant (e^C - 1)^2$$

[2] $\mathbf{c}\mathbf{B}^{-1}$ is rational but in typical cryptographic applications $\mathbf{c} \in \mathbb{Z}^n$ and $\mathcal{L}(\mathbf{B})$ is a q-ary lattice. Therefore, if carefully handled, $\mathbf{c}\mathbf{B}^{-1} \in \frac{1}{q}\mathbb{Z}^n$ can reasonably be expected to induce no significant imprecision.

Remark 3.17. The expression of the bound for the KL divergence can be simplified for practical parameters: if $\delta, \epsilon \ll 1$, then

$$e^C - 1 \approx C \approx \frac{8ns_1(\mathbf{C})\delta}{r}.$$

Proof. We first notice that $\|\mathbf{x} - \bar{\mathbf{x}}\| \leqslant \delta \cdot \|\mathbf{x}\|$. Let $\mathbf{y} = \mathbf{x} \cdot \mathbf{C}$ (resp. $\bar{\mathbf{y}} = \bar{\mathbf{x}} \cdot \bar{\mathbf{C}}$). We have

$$\bar{\mathbf{y}} = \mathbf{y} + (\bar{\mathbf{x}} - \mathbf{x})\mathbf{C} + \mathbf{x}(\bar{\mathbf{C}} - \mathbf{C}) + (\bar{\mathbf{x}} - \mathbf{x})(\bar{\mathbf{C}} - \mathbf{C})$$

From which it follows that

$$\|\bar{\mathbf{y}} - \mathbf{y}\| \quad \leqslant \quad \|\mathbf{x}\| s_1(\mathbf{C})(2\delta + \delta^2) \quad \leqslant 3\sqrt{n}\delta s_1(\mathbf{C}) \tag{3.9}$$

We now note $\rho_i \triangleq \rho_{r,y_i}$, $\bar{\rho}_i \triangleq \bar{\rho}_{r,\bar{y}_i}$, $\mathcal{D}_i \triangleq \lfloor y_i \rceil_r = \rho_i/\rho_i(\mathbb{Z})$, $\bar{\mathcal{D}}_i \triangleq \lfloor \bar{y}_i \rceil_r = \bar{\rho}_i/\bar{\rho}_i(\mathbb{Z})$ and $u_i(z) \triangleq \frac{(z-\bar{y}_i)^2}{2r^2} - \frac{(z-y_i)^2}{2r^2} = \frac{2z(y_i - \bar{y}_i) + (\bar{y}_i^2 - y_i^2)}{2r^2}$. For a given output $\hat{\mathbf{z}} = (\hat{z}_i)_i$ of PeikertSampler, we have:

$$\sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z \leftarrow \mathcal{D}_i}[u_i]\right] \leqslant \log\left(\frac{\mathcal{D}(\hat{\mathbf{z}})}{\bar{\mathcal{D}}(\hat{\mathbf{z}})}\right) \leqslant \sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z \leftarrow \bar{\mathcal{D}}_i}[u_i]\right] \tag{3.10}$$

We note A (resp B) the left (resp. right) term of equation 3.10. Bounding A and B will allow to conclude the proof by using Lemma 3.5.

$$\begin{aligned}
A &\triangleq \textstyle\sum_i \left[u_i(\hat{z}_i) - \mathbb{E}_{z_i \leftarrow \mathcal{D}_i}[u_i]\right] \\
&= \tfrac{1}{r^2} \textstyle\sum_i (y_i - \bar{y}_i)[(\hat{z}_i - y_i) - \mathbb{E}_{z_i \leftarrow \mathcal{D}_i}[z_i - y_i]] \\
\Rightarrow |A| &\leqslant \tfrac{1}{r^2} \sqrt{\textstyle\sum_i (y_i - \bar{y}_i)^2} \sqrt{\textstyle\sum_i ((\hat{z}_i - y_i) - \mathbb{E}_{z_i \leftarrow \mathcal{D}_i}[z_i - y_i])^2} \\
&\leqslant \tfrac{1}{r^2} \|\mathbf{y} - \bar{\mathbf{y}}\| \cdot \left[\|\hat{\mathbf{z}} - \mathbf{y}\| + \sqrt{\textstyle\sum_i \mathbb{E}_{z_i \leftarrow \mathcal{D}_i}[z_i - y_i]^2}\right] \\
&\leqslant \tfrac{1}{r^2} \cdot 3\sqrt{n}\delta s_1(\mathbf{C}) \cdot \left[r\sqrt{2\pi n} + \tfrac{r\sqrt{2\pi n}\epsilon}{1-\epsilon}\right] \\
&\leqslant \tfrac{8ns_1(\mathbf{C})\delta}{r} \cdot \left[1 + \tfrac{\epsilon}{1-\epsilon}\right]
\end{aligned} \tag{3.11}$$

The second equality develops the formula for each u_i, the third one uses the inequality of Cauchy-Schwartz: $|\langle \mathbf{u}, \mathbf{v} \rangle| \leqslant \|\mathbf{u}\| \cdot \|\mathbf{v}\|$. The fourth uses the triangle inequality: $\|\mathbf{u} + \mathbf{v}\| \leqslant \|\mathbf{u}\| + \|\mathbf{v}\|$. For the fifth one, we bound $\|\mathbf{y} - \bar{\mathbf{y}}\|$ using equation 3.9, assume without loss of generality that $\|\hat{\mathbf{z}} - \mathbf{y}\| \leqslant r\sqrt{2\pi n}$ using Lemma 3.9 and bound each $\mathbb{E}_{z_i \leftarrow \mathcal{D}_i}[z_i - y_i]$ using Lemma 3.8. The sixth and last equation simplifies the penultimate one. To bound B, we notice that

$$\begin{aligned}
B &= \tfrac{1}{r^2} \textstyle\sum_i (y_i - \bar{y}_i)[(\hat{z}_i - y_i) - \mathbb{E}_{z_i \leftarrow \bar{\mathcal{D}}_i}[z_i - \bar{y}_i] + (y_i - \bar{y}_i)] \\
\Rightarrow |B| &\leqslant \tfrac{8ns_1(\mathbf{C})\delta}{r} \cdot \left[1 + \tfrac{\epsilon}{1-\epsilon} + \tfrac{2s_1(\mathbf{C})\delta}{r}\right]
\end{aligned} \tag{3.12}$$

The bound on $|B|$ is obtained exactly as for $|A|$. We have $|A|, |B| \leqslant C$, which in turns allows to bound $\frac{\mathcal{D}}{\bar{\mathcal{D}}}$: $e^{-C} \leqslant \frac{\mathcal{D}}{\bar{\mathcal{D}}} \leqslant e^C$. We conclude using lemma 3.5. □

Corollary 3.18. *Let $n \geqslant 128$, $\epsilon \in (0,1)$, $r \geqslant \eta'_\epsilon(\mathbb{Z}^n)$, $\delta \in [0, .01)$ and $\mathbf{C}, \bar{\mathbf{C}} \in \mathbb{R}^{n \times n}$ be two matrices such that $s_1(\mathbf{C} - \bar{\mathbf{C}}) \leqslant \delta s_1(\mathbf{C})$. Let $\mathcal{D}$ (resp. $\bar{\mathcal{D}}$) be the output of* Peikert$(\mathbf{B}, \sqrt{\Sigma}, \mathbf{c})$

taking $\mathbf{C}$ *(resp.* $\bar{\mathbf{C}}$*) as a (possibly imprecise) precomputed value for* $\mathbf{C}$ *and calling* $\mathcal{N}_0^n$ *(resp.* $\mathcal{N}_\delta^n$*) as an oracle for* D_1^n*. Let:*

$$C = \frac{8ns_1(\mathbf{C})\delta}{r} \cdot \left[1 + \frac{\epsilon}{1-\epsilon} + \frac{2s_1(\mathbf{C})\delta}{r}\right]$$

The KL divergence between $\mathcal{D}$ *and* $\bar{\mathcal{D}}$ *is bounded as follows:*

$$D_{KL}(\bar{\mathcal{D}} \| \mathcal{D}) \leqslant (e^C - 1)^2$$

Proof. We can assume without loss of generality that D_1 samples a vector $\mathbf{x}$ of norm lesser than $\sqrt{2n}$. We fall in the conditions of Lemma 3.16, which gives the bound on the KL divergence.

□

Table 3.2: Precision for Peikert's Sampler

This table gives upper bounds on the precision sufficient for an imprecise Peikert's sampler to be indistinguishable (*i.e.* $D_{\mathrm{KL}} \leqslant 2^{-\lambda}$) from a perfect Peikert's sampler. Precisions are computed for $NTRU$ lattices of dimension $2N = 2048$, modulus $q = 1024$.

Security Level λ	Precision δ
80	60
128	84
192	116

3.4.4 A Note on Samplers over $\mathbb{Z}$

Our analyses presuppose we have access to perfect oracles for $\mathcal{N}(0,1)$ and $D_{\mathbb{Z},c,\sigma}$ for any $(c,\sigma) \in \mathbb{R} \times \mathbb{R}^+$. However, in practice one has to resort to approximate distributions. In both the continuous and discrete case, several methods exist, and depending on the goal one wishes to accomplish (low storage, software efficiency, hardware efficiency, etc.), one may use one or another.

- For continuous Gaussians, there exists a dizzyingly high number of methods. One of the most efficient, documented and implemented is arguably the *Ziggurat method* [MT00]. Other well-documented methods are Marsaglia's polar method [MB64] and the Box-Muller transform [BM58].
- Methods for sampling over discrete Gaussians are less numerous than their continuous counterparts. However, several methods have been proposed in the wake of lattice-based cryptography which consider using cumulative distribution tables [Pei10], lazy rejection sampling [DN12a], a discrete Ziggurat method [BCG+14], a Knuth-Yao based method focused on hardware implementations [RVV14] and an algorithm [DDLL13, Duc13, Lep14] developed for the BLISS signature scheme [DDLL13] but which can be adapted to other cases. A nice survey of these methods – and others – can be found in [DG14].

Each of these samplers has its own properties, so when implementing Klein's or Peikert's scheme in practice, these properties should be taken in account when considering the divergence of these samplers from perfect samplers.

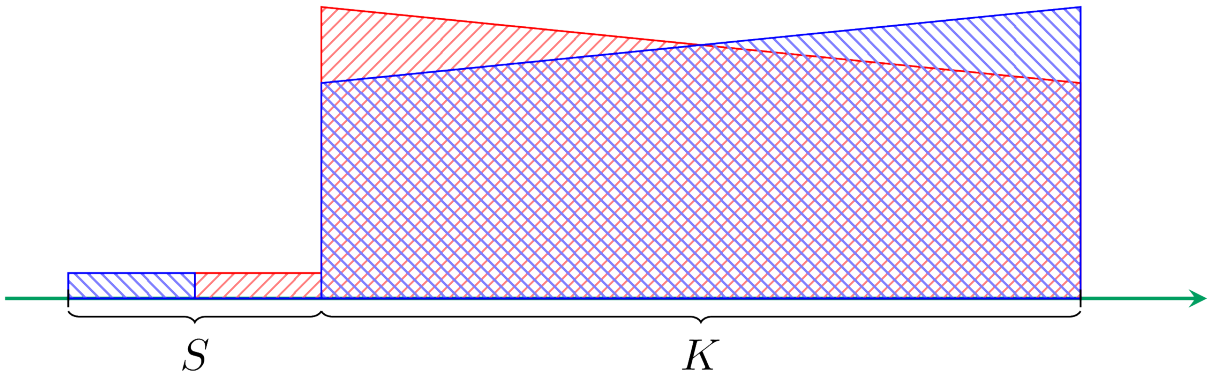


Figure 3.4: The statistical distance is more suited to analyze the distributions over S, and the KL divergence is better over K.

3.5 Conclusion

In this work, we showed that we can use the Kullback-Leibler divergence instead of the statistical distance as a metric to evaluate the security of schemes based on Gaussian samplers, and that there are at least two advantages in doing so. First, we can sample with a standard deviation about $\sqrt{2}$ smaller, second we can divide by about two the required precision. For 80 bits of security, we can even eliminate the use of high-precision numbers. This last statement assumes that the Gaussian samplers are run on a system that can process efficiently floating-point numbers with a 64-bits significand. Several common processors – such as IA32 (sometimes called i386), x86_64 and Itanium processors – already implement an 80-bit extended precision format with a 64-bit significand in hardware, and the trend seems to lean towards increased precision.[3]

It would be interesting to see if works obtained in this section could be improved by using other metrics. A promising metric is the Rényi divergence R_a, which has already found numerous applications in analyzing the security of lattice-based schemes. One could argue that the Rényi divergence is necessarily better than the KL divergence since the latter is just a particular case of the former for $a = 1$. Yet the Rényi divergence behaves completely differently when $a \neq 1$: in particular its probability preservation equation is not linear as for the statistical distance (equation 3.1) and the KL divergence (equation 3.2), but multiplicative.

A preliminary version of this work featured a "hybrid divergence" ∇_S defined as the statistical distance on a subset S of the support Ω, and as the KL divergence on its complement $\Omega \backslash S$. This approach was chosen to study pairs of distributions which are "statistical distance-close" on a part of their support, and "KL divergence-close" on another, such as the ones on Figure 3.4.

We later discarded this idea since it turned out to be unnecessary *in our case*. However, it would be interesting to see if cryptographic schemes (lattice-based or not) could benefit from it. Applying this "hybrid approach" to the Rényi divergence could prove interesting

[3] William Kahan, the primary architect behind the IEEE 754-1985 standard for floating-point computation, stated: "For now the 10-byte Extended format is a tolerable compromise between the value of extra-precise arithmetic and the price of implementing it to run fast; very soon two more bytes of precision will become tolerable, and ultimately a 16-byte format... That kind of gradual evolution towards wider precision was already in view when IEEE Standard 754 for Floating-Point Arithmetic was framed." [Hig02]

too as we would be able to combine not just two metrics, but an infinity, which could lead to tighter security proofs.

More generally, as lattice-based cryptography is gaining more and more maturity, the growing recourse to alternative and more *ad hoc* metrics than the statistical distance is a natural evolution of this field.

Chapter 4

Gaussian Sampling in Quasilinear Space over Structured Lattices

" This time it ain't just about being fast."

Dominic Toretto — Furious 7

4.1 Introduction

Sampling lattice points is one of the fundamental procedures in lattice cryptography. It is used in hash-and-sign signatures [GPV08], (hierarchical) identity-based encryption schemes [GPV08, CHKP10, ABB10b], standard-model signatures [ABB10b, Boy10b], attribute-based encryption [BGG+14], and many other constructions. Being able to output shorter vectors leads to more secure schemes, and the algorithm that produces the currently-shortest samples is the randomized version of Babai's nearest-plane algorithm [Bab85, Bab86] due to Klein [Kle00] and Gentry, Peikert, Vaikuntanathan [GPV08].

The main inefficiency of cryptography based on general lattices is that the key size is usually (at least) quadratic in the security parameter, which is related to the fact that a d-dimensional lattice is generated by d vectors. For security, the lattice dimension is usually taken to be on the order of 512, and this results in keys that are larger than one megabyte in size and unsuitable for most real-world applications. For this reason, all practical implementations of lattice schemes (e.g. [HPS98, LMPR08, LPR13a, DDLL13, DLP14]) rely on the hardness of problems involving polynomial rings [PR06, LM06, LPR13a], in which lattices can be represented by a few polynomials. Due to the fact that solving certain average-case problems over polynomial rings was shown to be as hard as solving worst-case problems in ideal lattices [SSTX09, LPR13a], building cryptographic systems using polynomial rings is often referred to as ideal lattice cryptography, even though ring lattice cryptography might be more accurate.

During its execution, the Klein/GPV algorithm is implicitly computing the Gram-Schmidt orthogonalization of the input basis.[1] Since the Gram-Schmidt procedure requires $\Theta(d^3)$ operations, the Klein/GPV sampler also requires at least this much time. For improving the time-complexity, one can pre-compute and store the Gram-Schmidt basis, which results in a sampling procedure that uses only $\Theta(d^2)$ operations. The Gram-Schmidt basis, however, requires the storage of $\Theta(d^2)$ elements, and so the key size is, as mentioned above, unacceptably large. One may hope that, again, using polynomials results in a decrease of the required storage. In this case, unfortunately, such rings do

[1] We recall that the Gram-Schmidt procedure produces a pairwise-orthogonal set of vectors $(\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_d)$ that span the same inner-product space as the input vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$

not help. The Gram-Schmidt orthogonalization procedure completely destroys the nice structure of polynomial lattices. So while a polynomial lattice basis can be represented by a few vectors, its Gram-Schmidt basis will have d vectors. Thus the $\Theta(d^2)$-operation Klein/GPV algorithm requires as much storage when using polynomial lattices as when using general lattices, and is equally unsuitable for practical purposes. Therefore the only realistic solution is to not store the pre-processed Gram-Schmidt basis, which would then allow for the ring lattice algorithm to be linear-space (since only the original, compact-representation basis needs to be stored), but require at least $\Omega(d^3)$ time due to the fact that the Gram-Schmidt basis will need to be computed.

4.1.1 Our Results

Our main result is an algorithm that computes the Gram-Schmidt basis of certain ring lattices using $\Theta(d^2)$, instead of $\Theta(d^3)$, arithmetic operations. We then show how this new procedure can be combined with Klein's sampler to achieve a "best-of-both-worlds" result – a sampling algorithm that requires $\Theta(d^2)$ operations, which does not require storing a pre-processed Gram-Schmidt basis. In ideal lattice cryptography, this implies being able to have keys that consist of just the compact algebraic basis requiring only linear storage. Not pre-computing the Gram-Schmidt basis of course necessarily slows down our sampling algorithm versus the one where this basis is already stored in memory. But our new orthogonalization algorithm is rather efficient, and the slowdown in some practical applications is by less than a factor of 4 (see the Table 4.2 in Section 4.9). In case of amortization (i.e. sampling more than one vector at a time), the running time of our new algorithm becomes essentially the same as that of the one requiring the storage of the orthogonalized basis. As a side note, since Klein's sampler is just a randomized version of the classic Babai nearest plane algorithm [Bab85, Bab86], all our improvements apply to the latter as well.

While analyzing the running-time of lattice algorithms, it is very important to not only consider the number of arithmetic operations, but also the arithmetic precision required for the algorithms to be stable. The run-time of algorithms that are not numerically stable may suffer due to the high precision required during their execution. A second important issue is understanding how the precision affects the statistical closeness of the distribution of the outputted vectors versus the desired distribution. We rigorously show that in order to have statistical closeness between the output distribution and the desired discrete Gaussian one be at most $2^{-\lambda}$, the required precision of the Gram-Schmidt basis needs to be a little more (in practice, less than a hundred bits) than λ. We then experimentally show that our new Gram-Schmidt procedure is rather numerically stable, and the intermediate storage is not much more than the final required precision. A third issue that also needs to be considered in practice is the space requirements of the algorithm during run-time. While the stored basis is very short, it could be that the intermediate computations require much larger storage (e.g. if the intermediate computation requires storing the entire Gram-Schmidt basis to a high precision). Our sampling algorithm, however, is rather efficient in this regard because it only requires storing one Gram-Schmidt vector at a time. The storage requirement during run-time is therefore less than 64KB for typical dimensions used in cryptographic applications.

4.1.1.1 Isometries and Ideal Lattices

Interestingly, our improved orthogonalization algorithm for polynomial lattices does not have much to do with their algebraic structure, but rather relies on their implicit geometric properties. The types of bases whose Gram-Schmidt orthogonalization we speed up are those that consist of a set of vectors that are related via a linear isometry. In particular, if H is a d-dimensional Hermitian inner-product space and $r : H \to H$ is a linear map that preserves the norm and is computable using $\mathcal{O}(d)$ operations, then we show (both theoretically and via implementations) that orthogonalizing a set of vectors $\{\mathbf{b}, r(\mathbf{b}), r^2(\mathbf{b}) \ldots, r^{d-1}(\mathbf{b})\}$ can be done using $\Theta(d^2)$ floating point operations.

4.1.1.2 Chapter Organization

In Section 4.2, we set up the notations and definitions that will be used throughout the chapter. In Section 4.3.1, we describe a simple version of our new algorithm that efficiently orthogonalizes a given set of vectors, and in Section 4.3.2 we give the full, optimized algorithm. In Section 4.4, we describe an algorithm that, given the orthogonalization, returns the transformation matrix μ that converts the set $\{\mathbf{b}, r(\mathbf{b}), \ldots, r(\mathbf{b})\}$ to $\{\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, \ldots, \tilde{\mathbf{b}}_n\}$. In Section 4.5, we extend our basic algorithms to those that can more efficiently orthogonalize sets of vectors of the form $\mathbf{b}_1, r(\mathbf{b}_1), \ldots, r^{n-1}(\mathbf{b}_1), \mathbf{b}_2, r(\mathbf{b}_2), \ldots, r^{n-1}(\mathbf{b}_2), \mathbf{b}_3, r(\mathbf{b}_3), \ldots$. These types of sets are the ones that normally occur as secret keys in lattice cryptography. A particular example of such a set is the NTRU lattice, which we discuss in Section 4.7. In that section, we also give timing comparisons between the exact version of our orthogonalization algorithm (which is analyzed in Section 4.6), and that of [GHN06], for computing the Gram-Schmidt orthogonalization of NTRU lattices. Since the two algorithms use different techniques to achieve speed-ups, we demonstrate that the two improvements can complement each other in the form of an even faster algorithm. In Section 4.8, we show how to implement Babai's nearest plane algorithm and the Klein/GPV sampling in linear space for lattices whose basis contains vectors that are related via an isometry. In Section 4.9 we focus on the implementation aspects of our results. In particular, we analyze the required precision to insure the correct functionality of our sampling algorithm.

4.1.2 Related Work

Toeplitz Orthogonalization. Computing faster orthogonalization for vectors that are somehow related has been considered in the past. For example, Sweet [Swe84] demonstrated an algorithm that orthogonalizes $d \times d$ Toeplitz matrices using $O(d^2)$ operations.

One may imagine that it may be possible to somehow adapt the results of [Swe84] to the orthogonalization of bases of ideal lattices. The idea would be to embed elements of $F = \mathbb{Q}[X]/\langle \Phi_m(X) \rangle$ into $C = \mathbb{Q}[X]/\langle X^m - 1 \rangle$ and then try to use the fact that in the coefficient representation, the elements $\mathbf{b}, \mathbf{b}X, \mathbf{b}X^2, \ldots$ in C form a Toeplitz matrix. One would have to also take care to make sure that the norm in F corresponds to the coefficient norm in C. We are not sure whether this direction is viable, and even if it is, the algorithm would necessarily involve computations over dimension m, rather than $\phi(m)$, which would unnecessarily increase the running-time of all algorithms.

Symplectic Orthogonalization. For the special case of NTRU lattices, Gama, Howgrave-Graham, and Nguyen [GHN06] devised algorithms that take advantage of a structure of NTRU bases called *symplecticity*. This allows them to be faster (by a constant factor)

than standard Gram-Schmidt orthogonalization when performing orthogonalization in exact arithmetic. We adapt our algorithms for the same application and they outperform those from [GHN06].[2] And since our algorithm and the one of [GHN06] rely on different ideas, it turns out that we can combine the two techniques to achieve a greater overall improvement (see Table 4.1 in Section 4.7).

Levinson Recursion. In a previous version of this work, we incorrectly assumed the core algorithm that we use throughout this chapter to be new. In fact, it is equivalent to a variant of the Levinson recursion performing the Arnoldi iteration for isometric operators. This is developed in Section 4.1.3.

4.1.3 Equivalence with Levinson Recursion

The Arnoldi iteration [Arn51] orthogonalizes a set of vectors $\{\mathbf{b}, f(\mathbf{b}), ...f^{n-1}(\mathbf{b})\}$ where f is a linear operator.[3] Lanczos showed [Lan50] that when f is Hermitian, this process could be sped up by a factor $O(n)$. In Section 4.3, we do the same as Lanczos, but for f isometric. The similarity of our setting with Lanczos' led us to questioning whether such an algorithm existed.

On the other hand, the Levinson recursion was discovered by Levinson [Lev47] and improved by Durbin [Dur60]. This algorithm (sometimes called Levinson-Durbin recursion) solves linear systems $\mathbf{y} = \mathbf{xM}$ in time $O(n^2)$ when $M \in \mathbb{R}^{n\times n}$ is a Toeplitz matrix, instead of time $O(n^3)$ for generic methods like Gauss-Jordan elimination.

Gragg showed [Gra93] that Levinson recursion could be interpreted differently to give an equivalent to Lanczos' algorithm, but for f isometric, which is exactly what we do in Section 4.3. Therefore, this section may not be considered as new material, however we keep it in its current state as we believe it helps the comprehension and self-containedness of this chapter.

Moreover, to the best of of knowledge, this work (more precisely, the article [LP15] co-written with Vadim Lyubashevsky) is the first to discuss the applications of the Levinson recursion to lattice-based cryptography, as well as its exact complexity.

We thank Gilles Villard for pointing us to Gragg's paper.

4.2 Preliminaries

4.2.1 Notations

We recall that most of the notation conventions are stated in pages 11 to 19. Throughout the chapter, we will be working over a d-dimensional inner product space H (usually $H = \mathbb{R}^d$ or $\mathbb{C}^d$), with $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ being a scalar product and the associated norm over H. Except when stated otherwise, vectors will be written in bold, matrices and bases in uppercase bold, and scalars in non-bold letters. $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ will be either an ordered set of independent vectors, also called a basis, or the $n \times d$ matrix whose rows are the $\mathbf{b}_i$. We also note $\mathbf{B}_k \triangleq \{\mathbf{b}_1, ..., \mathbf{b}_k\}$.

[2] We mention that [GHN06] also contains other results which are independent of Gram-Schmidt orthogonalization and are therefore not improved by our work.

[3] The subspaces generated by such sets of vectors are often called *Krylov Subspaces* $\mathcal{K}_n(f, \mathbf{b})$.

Definition 4.1. *A linear isometry is a linear map $r : H \to H$ such that for any $\mathbf{x}, \mathbf{y} \in H$:*

$$\langle r(\mathbf{x}), r(\mathbf{y}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle,$$

or equivalently

$$\|\mathbf{x}\| = \|r(\mathbf{x})\|.$$

For conciseness, we will sometimes say isometry instead of linear isometry. Since the dimension of H is finite, it is immediate that r is invertible. We will be assuming throughout the work that both r and r^{-1} are computable in time $\mathcal{O}(d)$.

We recall that $\mathbf{Proj}(\mathbf{x}, F)$ is the orthogonal projection of $\mathbf{x}$ over F. We have the following properties.

Proposition 4.2. *Let r be an isometry and and F be a subspace of H. Then:*
1. $\mathbf{x} \perp F \Rightarrow r(\mathbf{x}) \perp r(F)$
2. $r(\mathbf{Proj}(\mathbf{x}, F)) = \mathbf{Proj}(r(\mathbf{x}), r(F))$

Proof. We prove the two claims separately:
1. Since r preserves the dot product, it also preserves orthogonality between vectors.
2. r preserves the norm, so

$$\|\mathbf{x} - \mathbf{y}\| = \min_{\mathbf{z} \in F} \|\mathbf{x} - \mathbf{z}\| \Longrightarrow \|r(\mathbf{x}) - r(\mathbf{y})\| = \min_{\mathbf{z} \in r(F)} \|r(\mathbf{x}) - \mathbf{z}\|$$

□

4.2.2 The Gram-Schmidt Orthogonalization

We recall that the Gram-Schmidt orthogonalization (GSO) of a basis $\mathbf{B} = \{\mathbf{b}_i, ..., \mathbf{b}_n\}$ is the unique basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$ verifying one of these properties:

- $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \mathbf{Proj}(\mathbf{b}_k, \text{Span}(\mathbf{B}_{k-1}))$
- $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{b}_k, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j$
- $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k \perp \text{Span}(\mathbf{B}_{k-1})$ and $(\mathbf{b}_k - \tilde{\mathbf{b}}_k) \in \text{Span}(\mathbf{B}_{k-1})$

For each k, $\tilde{\mathbf{b}}_k$ is the Gram-Schmidt reduction (GSR) of $\mathbf{b}_k$. For random bases, the most commonly known algorithm for computing their GSO is the Gram-Schmidt process, which is given in Section 2.1.3. Other known methods rely either on Givens rotations, Householder matrices or Cholesky decomposition [GVL96, Ste98] are are often preferred in practice for stability reasons [Ste10]. However, they all achieve complexity $\Theta(n^2 d)$ (or $\Theta(n^3)$ for those who can be applied only when $n = d$).

4.2.3 The Gram-Schmidt Decomposition

The Gram-Schmidt decomposition (GSD) is a natural by-product of the Gram-Schmidt orthogonalization which gives the relation between the input and output bases in the form of a matrix $\mathbf{L}$. Such a matrix is useful in many cases – for example, its role is critical in the LLL algorithm [LLL82]. Outside cryptography, applications include solving undetermined linear systems, least square problems and computing eigenvalues.

Definition 4.3 (Gram-Schmidt Decomposition)**.** *Let* $\mathbf{B}$ *be a* $d \times n$ *matrix.* $\mathbf{B}$ *can be uniquely decomposed as* $\mathbf{B} = \mathbf{L} \times \tilde{\mathbf{B}}$*, where* $\tilde{\mathbf{B}}$ *is the GSO of* $\mathbf{B}$ *and* $\mathbf{L} = (L_{i,j})_{1 \leqslant i,j \leqslant n}$ *is the lower triangular matrix such that*

$$L_{i,j} = \begin{cases} \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} & \text{if } i > j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This is called the Gram-Schmidt decomposition (or GSD) of $\mathbf{B}$*.*

Notice that the matrix $\mathbf{L}$ is automatically constructed in Algorithm 2.1 while computing the GSO. This, however, will not be the case in our improved GSO algorithm, and this is why in this chapter we will differentiate between GSO and GSD.

4.2.4 Anticirculant Matrices and Isometries

In this section, we define anticirculant matrices and explain the geometric properties that bound their row vectors. We first give the definitions of cyclotomic elements.

Cyclotomic Polynomials, Rings and Fields

Definition 4.4. *Let* $m \in \mathbb{N}^\star$ *and* ζ_m *denote any primitive* m*-th root of unity in* $\mathbb{C}$*, per example* $\zeta_m = e^{\frac{2i\pi}{m}}$*. We note* $\Omega_m = \{\zeta_m^k, k \in \mathbb{Z}_m^\times\}$ *the set of primitive* m*-th roots of unity. We also note* ϕ_m *and call* m*-th cyclotomic polynomial the polynomial defined by*

$$\forall x \in \mathbb{C}, \phi_m(x) = \prod_{\zeta \in \Omega_m} (x - \zeta) = \prod_{k \in \mathbb{Z}_m^\times} (x - \zeta_m^k)$$

We call cyclotomic field associated to ϕ_m *the number field* $\mathbb{Q}(\zeta_m)$ *obtained by adjoining* ζ_m *to* $\mathbb{Q}$*. We also note call cyclotomic ring associated to* ϕ_m *the ring* $\mathbb{Z}[\zeta_m]$*.*

For some values of m – which are also the vast majority of values encountered in lattice-based cryptography –, ϕ_m is easy to compute:

- For a prime p, $\phi_p(x) = \sum_{k=0}^{p-1} x^k$
- For a prime power $m = p^k$, $\phi_m(x) = \phi_p(x^{m/p})$
- If $m = 2^k$ is a power of two, $\phi_m(x) = x^{m/2} + 1$

We now cite a few useful properties of cyclotomic polynomials, rings and fields. $\mathbb{Z}[\zeta_m]$ is the ring of integers of $\mathbb{Q}(\zeta_m)$, and conversely $\mathbb{Q}(\zeta_m)$ is the field of fractions of $\mathbb{Z}[\zeta_m]$.[4] The field $\mathbb{Q}(\zeta_m)$ (resp. the ring $\mathbb{Z}[\zeta_m]$) is isomorphic to $\mathbb{Q}[x]/(\phi_m(x))$ (resp. $\mathbb{Z}[x]/(\phi_m(x))$). The polynomial ϕ_m has integer coefficients. Noting $\mathbb{Z}_m^\times$ the group of invertible elements of $\mathbb{Z}_m$, the degree of ϕ_m is $\varphi(m)$, where $\varphi(m) \stackrel{\Delta}{=} |\mathbb{Z}_m^\times|$ denotes Euler's totient function evaluated on m. One can show that

$$x^d - 1 = \prod_{m|d} \phi_m(x) \tag{4.1}$$

which is a fact that we will use in Chapter 7, and is useful when working with cyclotomic fields. Indeed, if p is prime, then it is much more practical to work in $\mathbb{Z}[x]/(x^p - 1)$ rather than in $\mathbb{Z}[x]/(\phi_p(x))$.

[4] This correspondence is not true in general.

Anticirculant Matrices

In lattice-based cryptography, it is common to work with matrices which are structured over cyclotomic rings, as defined hereunder.

Definition 4.5. *For any $n \in \mathbb{N}^\star$, ϕ a monic polynomial with n distinct roots over $\mathbb{C}$ and $f \in \mathbb{R}[x]$, we note $\mathcal{A}_\phi(f)$ the $n \times n$ matrix defined whose i-th row are the coefficients of $x^{i-1} \cdot f \bmod \phi$:*

$$\mathcal{A}_\phi(f) \triangleq \begin{bmatrix} f \bmod \phi \\ x \cdot f \bmod \phi \\ \vdots \\ x^{n-1} \cdot f \bmod \phi \end{bmatrix}$$

When ϕ is clear from context, we note $\mathcal{A}(f) = \mathcal{A}_\phi(f)$.

We notice that if $f \neq 0$ and ϕ is irreductible, then $\mathcal{A}_\phi(f)$ is invertible and its row vectors form a basis. We abusively call "anticirculant" the matrices of the form $\mathcal{A}_\phi(f)$, because they generalize circulant matrices: in particular, for $m = 2^k$, $\mathcal{A}_{\phi_m}(f)$ is a signed variant of a circulant matrix. Just like their circulant counterparts, anticirculant matrices have very useful properties. The most important one is that for a fixed ϕ, all the matrices $\mathcal{A}_\phi(f)$ are co-diagonalizable:

$$\mathcal{A}_\phi(f) = \mathbf{V}^{-1} \cdot \mathrm{diag}(\{f(\omega)\}_{\phi(\omega)=0}) \cdot \mathbf{V}$$

where $\mathbf{V}$ is the Vandermonde matrix associated to the roots ω of ϕ. As an immediate consequence, the set $\mathcal{A}_\phi \triangleq \{\mathcal{A}_\phi(f) | f \in \mathbb{R}[x]\}$ is a commutative ring.

Link with Ideal Lattices and Isometries

We now explain the connection between isometries, ideal lattices and anticirculant matrices. Consider $F = \mathbb{R}[x]/(\phi_m(x))$. Elements in F can be represented via a canonical embedding[5] into $\mathbb{C}^{\phi(m)}$, and in that case F becomes isomorphic, as an inner product space, to $\mathbb{R}^{\phi(m)}$ where the inner product $\langle \mathbf{a}, \mathbf{b} \rangle$ of $\mathbf{a}, \mathbf{b} \in F$ is defined as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{\zeta \in \Omega_m} \mathbf{a}(\zeta) \cdot \mathbf{b}(\zeta) = \sum_{i \in \mathbb{Z}_m^\times} \mathbf{a}(\zeta_m^i) \cdot \mathbf{b}(\zeta_m^i),$$

where $\zeta_m \in \mathbb{C}$ is a primitive m^{th} root of unity (c.f. [LPR13b, Sections 2.2, 2.5.2]).[6]

With the above definition of inner product (which is in fact the usual inner product over $\mathbb{C}^{\phi(m)}$ when elements in F are represented via the canonical embedding), the norm of an element $\mathbf{b} \in F$ is

$$\|\mathbf{b}\| = \sqrt{\sum_{\zeta \in \Omega_m} |\mathbf{b}(\zeta)|^2}.$$

[5] The canonical embedding of a polynomial $\mathbf{b} \in F$ is a vector in $\mathbb{C}^{\phi(m)}$ whose coefficients are the evaluations of $\mathbf{b}$ on each of the $\phi(m)$ complex roots of $\Phi_m(X)$. Due to the fact that half of the roots of $\Phi_m(X)$ are conjugates of the other half, the resulting embedded vector in $\mathbb{C}^{\phi(m)}$ can be represented by $\phi(m)$ real numbers.

[6] We point out that the actual computation of the inner product does not require any operations over $\mathbb{C}$. The reason is that $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i \in \mathbb{Z}_m^\times} \mathbf{a}(\zeta_m^i) \cdot \mathbf{b}(\zeta_m^i)$ can be rewritten as $(\mathbf{Va})^\star \mathbf{Vb} = \mathbf{a}^\star \mathbf{V}^\star \mathbf{Vb}$ for a Vandermonde matrix $\mathbf{V}$ with coefficients in $\mathbb{C}$. The matrix $\mathbf{V}^\star \mathbf{V}$, however, is a simple integer matrix, multiplication by which can be performed in linear time for most "interesting" cyclotomic polynomials (e.g. m is prime or a power of 2).

One can check that $\|\mathbf{b}\| = \|x \cdot \mathbf{b}\|$. Since the function $r : F \to F$ defined as $r(\mathbf{b}) = x \cdot \mathbf{b}$ is linear, it is also an isometry (since it preserves the norm). Furthermore, since F is a field, for any non-zero $\mathbf{b} \in F$, the rows $\mathbf{b}, x\mathbf{b}, x^2\mathbf{b}, \ldots, x^{\phi(m)-1}\mathbf{b}$ of $\mathcal{A}_{\phi_m}(\mathbf{b})$ are all linearly-independent. When $\mathbf{b}$ is an element of $\mathcal{R} = \mathbb{Z}[x]/(\phi_m(x))$, the set

$$\{\mathbf{b}, x\mathbf{b}, x^2\mathbf{b}, \ldots, x^{\phi(m)-1}\mathbf{b}\} = \{\mathbf{b}, r(\mathbf{b}), r^2(\mathbf{b}), \ldots, r^{\phi(m)-1}(\mathbf{b})\}$$

therefore generates the ideal $\langle \mathbf{b} \rangle$ as an additive group. Such bases containing short elements can serve as private keys in cryptographic schemes.[7]

4.3 Gram-Schmidt Orthogonalization over Isometric Bases

In this section we present our improved isometric Gram-Schmidt algorithm. In Section 4.3.1, we present a first simple version which we believe is very intuitive to understand, and then present a slightly faster, more involved, version of it in Section 4.3.2.

Definition 4.6. *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ *be an ordered basis of a lattice* $\Lambda \subseteq H$*. We say that* $\mathbf{B}$ *is isometric if there exists an isometry* r *such that*

$$\forall k \in [\![2, n]\!], \mathbf{b}_k = r(\mathbf{b}_{k-1})$$

4.3.1 A Quadratic-Time algorithm

We now describe a simple algorithm that computes the GSO of any isometric basis in time $\Theta(nd)$ (or $\Theta(n^2)$ when $n = d$).

We briefly expose the general idea behind the algorithm before presenting it formally. If $\tilde{\mathbf{b}}_k$ is the GSR of $\mathbf{b}_k$, then $r(\tilde{\mathbf{b}}_k)$ is almost the GSR of $\mathbf{b}_{k+1}$: it is orthogonal to $\mathbf{b}_2, ..., \mathbf{b}_k$, but not to $\mathbf{b}_1$. However, reducing $r(\tilde{\mathbf{b}}_k)$ with respect to $\mathbf{b}_1$ would break its orthogonality to $\mathbf{b}_2, ..., \mathbf{b}_k$, so what we really need to do is to reduce it with respect to $\mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, \mathrm{Span}(\mathbf{b}_2, ..., \mathbf{b}_k))$. Indeed, this latter vector is orthogonal to $\mathbf{b}_2, ..., \mathbf{b}_k$, so reducing $r(\tilde{\mathbf{b}}_k)$ with respect to it won't break the orthogonality of $r(\tilde{\mathbf{b}}_k)$ to $\mathbf{b}_2, ..., \mathbf{b}_k$. Fortunately, $\mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, \mathrm{Span}(\mathbf{b}_2, ..., \mathbf{b}_k))$ can itself be updated quickly. Definition 4.7 and Algorithm 4.1 formalize this idea.

Definition 4.7. *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ *be an ordered basis and* $k \in [\![1, n]\!]$*. We denote* $\mathbf{v}_{\mathbf{B},k} = \mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, r(\mathrm{Span}(\mathbf{B}_{k-1})))$*. When* $\mathbf{B}$ *is obvious from the context, we simply write* $\mathbf{v}_k$*.*

Proposition 4.8. *Let* $\mathbf{B}$ *be an isometric basis with respect to* r*. Algorithm 4.1 returns the GSO of* $\mathbf{B}$*. Moreover, if* $r(\mathbf{v})$ *can be computed in time* $O(d)$ *for any* $\mathbf{v} \in H$*, then Algorithm 4.1 terminates in time* $O(nd)$*.*

Proof. We first prove the correctness of the scheme by proving by induction that for every $k \in [\![1, n]\!]$, we have the following:

[7] Normally, the bases used in schemes have slightly different forms, such as consisting of a concatenation of elements from $\mathcal{R}$, or being formed by several elements in $\mathcal{R}$. Such bases still contain large components that are related via an isometry, and we discuss this in more detail in Sections 4.5 and 4.7.

Algorithm 4.1 IsometricGSO($\mathbf{B}$)

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$
Ensure: Gram-Schmidt reduced basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$
1: $\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$
2: $\mathbf{v}_1 \leftarrow \mathbf{b}_1$
3: **for** $k = 1, ..., n-1$ **do**
4: $\quad \tilde{\mathbf{b}}_{k+1} \leftarrow r(\tilde{\mathbf{b}}_k) - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle}{\|\mathbf{v}_k\|^2}\mathbf{v}_k$
5: $\quad \mathbf{v}_{k+1} \leftarrow \mathbf{v}_k - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle}{\|\tilde{\mathbf{b}}_k\|^2} r(\tilde{\mathbf{b}}_k)$
6: **end for**
7: **return** $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

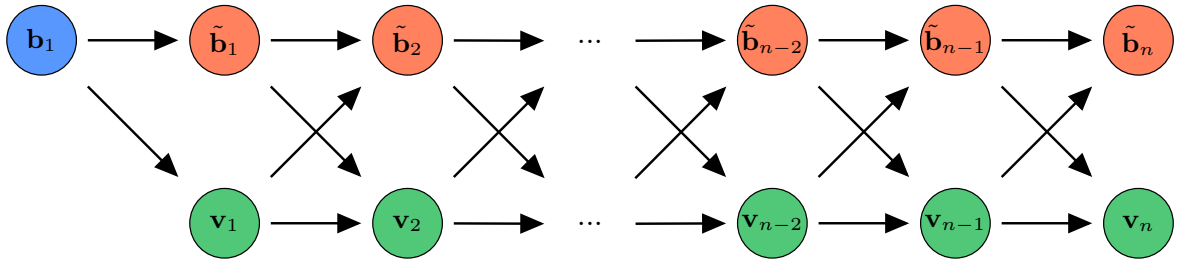


Figure 4.1: Computing all the orthogonalized vectors from the first one in Algorithm 4.1

- $\mathbf{v}_k = \mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, r(\text{Span}(\mathbf{B}_{k-1})))$ (1)
- $\tilde{\mathbf{b}}_k = \mathbf{b}_k - \mathbf{Proj}(\mathbf{b}_k, \text{Span}(\mathbf{B}_{k-1}))$ (2)

This is trivially true for $k = 1$. Assuming (1) and (2) are true at step k, we have:

- Since $\mathbf{v}_k$ and $\tilde{\mathbf{b}}_k$ are already orthogonal to $r(\text{Span}(\mathbf{B}_{k-1}))$, $\mathbf{v}_{k+1}$ also is as a linear combination of the two. But $\mathbf{v}_{k+1}$ is also the orthogonalization of $\mathbf{v}_k$ w.r.t. $r(\tilde{\mathbf{b}}_k)$, so it is orthogonal to $r(\text{Span}(\mathbf{B}_{k-1})) + \text{Span}(r(\tilde{\mathbf{b}}_k)) = r(\text{Span}(\mathbf{B}_k))$. On the other hand, $\mathbf{b}_1 - \mathbf{v}_k$ is in $r(\text{Span}(\mathbf{B}_{k-1}))$ so $\mathbf{b}_1 - \mathbf{v}_{k+1}$ is in $r(\text{Span}(\mathbf{B}_k))$. By applying Definition 2.19, we can conclude that (1) is true for $k+1$.

- The same reasoning holds for $\tilde{\mathbf{b}}_{k+1}$: it is orthogonal to $r(\text{Span}(\mathbf{B}_{k-1}))$ because both $\mathbf{v}_k$ and $r(\tilde{\mathbf{b}}_k)$ are. But since it also is orthogonalized w.r.t. $\mathbf{v}_k$ (in line 4 of the algorithm), it then is orthogonal to $r(\text{Span}(\mathbf{B}_{k-1})) + \text{Span}(\mathbf{v}_k) = \text{Span}(\mathbf{B}_k)$. On the other hand, $\mathbf{b}_{k+1} - \tilde{\mathbf{b}}_{k+1} = r(\mathbf{b}_k - \tilde{\mathbf{b}}_k) + \frac{\langle r(\tilde{\mathbf{b}}_k), \mathbf{v}_k\rangle}{\langle \mathbf{v}_k, \mathbf{v}_k\rangle}\mathbf{v}_k$ is in $\text{Span}(\mathbf{B}_k)$. As before, we can conclude that (2) is true for $k+1$.

Since (2) is verified for any $k \in [\![1, n]\!]$, $\tilde{\mathbf{B}}$ is the GSO of $\mathbf{B}$.

The time complexity of the algorithm is straightforward: assuming additions, subtractions, multiplications and divisions are done in constant time, each scalar product or square norm takes time $\mathcal{O}(d)$. Since there are $3(n-1)$ norms or scalar products, and $2(n-1)$ computations of $r(.)$, the total complexity is $\mathcal{O}(nd)$. □

4.3.2 Making Isometric GSO Faster

Algorithm 4.1 is already $O(n)$ times faster than the classical Gram-Schmidt process. In this subsection, we show that intermediate values are strongly interdependent and that this fact can be used to speed up our GSO implementation by about 67%.

Lemma 4.9. *Let* $\mathbf{B}$ *be an isometric basis. For any* k *in* $[\![1, n]\!]$*, we have the following equalities:*
- $\langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_k)\rangle = \langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$
- $\|\mathbf{v}_k\|^2 = \langle \mathbf{v}_k, \mathbf{v}_1\rangle = \|\tilde{\mathbf{b}}_k\|^2$

When implicit from context, we will denote $C_k = \langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$ *and* $D_k = \|\tilde{\mathbf{b}}_k\|^2$*. We have the following recursive formula:*

$$\forall k \in [\![1, n-1]\!], D_{k+1} = D_k - \frac{C_k^2}{D_k}$$

Proof. We prove each of the three equalities separately:
- The equality $\langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_k)\rangle = \langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$ is equivalent to $\langle \mathbf{v}_k - \mathbf{v}_1, r(\tilde{\mathbf{b}}_k)\rangle = 0$, which is true since $\mathbf{v}_k - \mathbf{v}_1 = \mathbf{Proj}(\mathbf{b}_1, r(\mathrm{Span}(\mathbf{B}_{k-1})))$ is in the subspace $r(\mathrm{Span}(\mathbf{B}_{k-1}))$ and $\tilde{\mathbf{b}}_k$ is orthogonal to $r(\mathrm{Span}(\mathbf{B}_{k-1}))$
- The equality $\|\mathbf{v}_k\|^2 = \langle \mathbf{v}_k, \mathbf{v}_1\rangle$ is obtained by following the same reasoning as above
- The equality $\|\mathbf{v}_k\|^2 = \|\tilde{\mathbf{b}}_k\|^2$ is shown by induction: it is the case for $k = 1$. By observing that $\tilde{\mathbf{b}}_{k+1}$ is orthogonal to $\mathbf{v}_k$ from line 4 of Algorithm 4.1(resp. $\mathbf{v}_{k+1}$ is orthogonal to $r(\mathbf{b}_k)$ from line 5)), we can use the Pythagorean theorem to compute $\|\tilde{\mathbf{b}}_{k+1}\|^2$ and $\|\mathbf{v}_{k+1}\|^2$:

$$\|\tilde{\mathbf{b}}_{k+1}\|^2 = \|\tilde{\mathbf{b}}_k\|^2 - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle^2}{\|\mathbf{v}_k\|^2} \quad \text{and} \quad \|\mathbf{v}_{k+1}\|^2 = \|\mathbf{v}_k\|^2 - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle^2}{\|\tilde{\mathbf{b}}_k\|^2}$$

  At which point we can conclude by induction that $\|\mathbf{v}_{k+1}\|^2 = \|\tilde{\mathbf{b}}_{k+1}\|^2$, and these equalities also yield the recursive formula $D_{k+1} = D_k - \frac{C_k^2}{D_k}$.

□

This result allows us to speed up further the GSO for isometric bases. At each iteration of the algorithm IsometricGSO, instead of computing $\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$, $\|\tilde{\mathbf{b}}_k\|^2$ and $\|\mathbf{v}_k\|^2$, one only needs to compute $\langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_k)\rangle$, and can instantly compute $\|\tilde{\mathbf{b}}_k\|^2 = \|\mathbf{v}_k\|^2$ from previously known values. We choose $\langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_k)\rangle$ rather than $\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$ because $\mathbf{v}_1$ has a much smaller bitsize than $\mathbf{v}_k$, resulting in a better complexity in exact arithmetic. Moreover, in the case where we use floating-point arithmetic, $\mathbf{v}_1$ does not introduce any floating-point error, unlike $\mathbf{v}_k$.
Algorithm 4.2 – which can be seen [Gra93] as equivalent to the Levinson Recursion [Lev47, Dur60]– sums up these enhancements.

Proposition 4.10. *If* $\mathbf{B}$ *is an isometric basis, then Algorithm 4.2 returns the GSO of* $\mathbf{B}$*. Moreover, if we disregard the computational cost of* r*, then Algorithm 4.2 performs essentially* $3n^2$ *multiplications (resp. additions), whereas Algorithm 4.1 performs essentially* $5n^2$ *multiplications (resp. additions).*

Proof. For the correctness of Algorithm 4.2, one only needs to show that at each step, $C_k = \langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k)\rangle$ and $D_k = \|\tilde{\mathbf{b}}_k\|^2 = \|\mathbf{v}_k\|^2$. The first and third equalities are given by lemma 4.9, and the second one by induction: assuming that C_k, D_k are correct, D_{k+1} is

Algorithm 4.2 FasterIsometricGSO($\mathbf{B}$)

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$
Ensure: Gram-Schmidt reduced basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}(, (C_k)_{1 \leqslant k < n}, (D_k)_{1 \leqslant k < n})$
1: $\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$
2: $\mathbf{v}_1 \leftarrow \mathbf{b}_1$
3: $C_1 \leftarrow \langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_1) \rangle$
4: $D_1 \leftarrow \|\mathbf{b}_1\|^2$
5: **for** $k = 1, ..., n-1$ **do**
6: $\quad \tilde{\mathbf{b}}_{k+1} \leftarrow r(\tilde{\mathbf{b}}_k) - \frac{C_k}{D_k} \mathbf{v}_k$
7: $\quad \mathbf{v}_{k+1} \leftarrow \mathbf{v}_k - \frac{C_k}{D_k} r(\tilde{\mathbf{b}}_k)$
8: $\quad C_{k+1} \leftarrow \langle \mathbf{v}_1, r(\tilde{\mathbf{b}}_{k+1}) \rangle$
9: $\quad D_{k+1} \leftarrow D_k - \frac{C_k^2}{D_k}$
10: **end for**
11: **return** $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

correct, once again from lemma 4.9.

□

4.4 Gram-Schmidt Decomposition over Isometric Bases

In this section, we show that the computation of the matrix $\mathbf{L}$ from the Gram-Schmidt decomposition (or GSD, see Definition 4.3) can be sped up by a $O(n)$ factor in the case of isometric matrices by using tricks similar to those which led to the speeding-up of GSO. The proof of the following theorem explains how to compute the GSD of an isometric basis/matrix in quadratic time.

Theorem 4.11. *Let $\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n)$ be an isometric basis and $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n)$ its GSO. For the sake of simplicity, we identify the basis $\mathbf{B}$ (resp. $\tilde{\mathbf{B}}$) to the (not necessarily square) matrix which rows are the vectors of the basis. Assume we already have $\mathbf{B}$ and $\tilde{\mathbf{B}}$, along with the values $C_j = \langle \mathbf{v}_j, r(\tilde{\mathbf{b}}_j) \rangle, D_j = \|\tilde{\mathbf{b}}_j\|^2$ for $1 \leqslant j < n$. Then the matrix $\mathbf{L}$ associated to $\mathbf{B}$ can be computed in time $O(n^2)$.*

Proof. For $1 \leqslant i < j \leqslant n$, let $X_{i,j} = \langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle$ and $Y_{i,j} = \langle r(\mathbf{b}_i), \mathbf{v}_j \rangle$. All the nontrivial values of $L_{i,j}$ (that is, the values $L_{i,j}$ for $1 \leqslant j < i \leqslant n$) can be expressed as $L_{i,j} = \frac{X_{i,j}}{D_j}$. The values $X_{i,j}, Y_{i,j}$ satisfy these recursive formulae:

$$\begin{cases} X_{i+1,j+1} = X_{i,j} - \frac{C_j}{D_j} Y_{i,j} \\ Y_{i,j+1} = Y_{i,j} - \frac{C_j}{D_j} X_{i,j} \end{cases}$$

These formulae allow us to compute all the values of $X_{i,j}, Y_{i,j}$ from the $2(n-1)$ values $X_{i,1}, Y_{i,1}$. Once all of these values are computed, one can simply obtain the $L_{i,j}$ from the $X_{i,j}$. Algorithm 4.3 puts this idea into practice.

The idea of this algorithm is somewhat similar to the one behind Algorithms 4.1 and 4.2: the only values that we really need to compute are the $X_{i,j}$'s, but in order to do that efficiently we resort to a mutual recursion involving the $Y_{i,j}$'s.

Algorithm 4.3 $\mathsf{IsometricGSD}(\mathbf{B}, \tilde{\mathbf{B}}, (C_i), (D_i))$

Require: Basis $\mathbf{B}$ and its orthogonalization $\tilde{\mathbf{B}}$, values $C_j = \langle \mathbf{v}_j, r(\tilde{\mathbf{b}}_j)\rangle, D_j = \|\tilde{\mathbf{b}}_j\|^2$ for $1 \leqslant j < n$
Ensure: Matrix $\mathbf{L} = \mathbf{B} \times \tilde{\mathbf{B}}^{-1}$
1: Set the diagonal values of $\mathbf{L}$ to 1 and the values above the diagonal to 0
2: **for** $i = 2...n$ **do** // Computing the $(X_{i,1}), (Y_{i,1})$
3: $X_{i,1} \leftarrow \langle \mathbf{b}_i, \tilde{\mathbf{b}}_1 \rangle$
4: $Y_{i,1} \leftarrow \langle r(\mathbf{b}_i), \mathbf{b}_1 \rangle$
5: **for** $j = 2...i-1$ **do**
6: $X_{i,j} \leftarrow X_{i-1,j-1} - \frac{C_{j-1}}{D_{j-1}} Y_{i-1,j-1}$
7: $Y_{i,j} \leftarrow Y_{i,j-1} - \frac{C_{j-1}}{D_{j-1}} X_{i,j-1}$
8: **end for**
9: **end for**
10: **for** $i = 2...n$ **do** // Filling out the non-trivial values of $L_{i,j}$
11: **for** $j = 1...i-1$ **do**
12: $L_{i,j} \leftarrow \frac{X_{i,j}}{D_j}$
13: **end for**
14: **end for**
15: **return** $\mathbf{L} = (L_{i,j})_{1 \leqslant i,j \leqslant n}$

The time complexity is straightforward. Each $X_{i,j}, Y_{i,j}$ takes time $O(1)$ to be computed, except for $2n$ of them which need time $O(n)$ each. So the overall cost is $O(n^2)$. □

As an example, the matrices X_{steps} and X_{chrono} below show, for $n = 5$, in which order the matrices X, Y are filled. The two matrices use different metrics: X_{chrono} displays the chronological order in which the matrices are filled by the algorithm, whereas X_{steps} display the minimal depth of the computational tree necessary in order to compute an $X_{i,j}$ (resp. $Y_{i,j}$). If a box contains $\times$, it means that the corresponding value is trivial (see step 1 of the algorithm).

$$X_{steps} = \begin{array}{|c|c|c|c|c|}\hline \times & \times & \times & \times & \times \\ \hline 1 & \times & \times & \times & \times \\ \hline 1 & 2 & \times & \times & \times \\ \hline 1 & 2 & 3 & \times & \times \\ \hline 1 & 2 & 3 & 4 & \times \\ \hline \end{array} \qquad X_{chrono} = \begin{array}{|c|c|c|c|c|}\hline \times & \times & \times & \times & \times \\ \hline 1 & \times & \times & \times & \times \\ \hline 2 & 3 & \times & \times & \times \\ \hline 4 & 5 & 6 & \times & \times \\ \hline 7 & 8 & 9 & 10 & \times \\ \hline \end{array}$$

As X_{chrono} shows, the algorithm fills the matrices X, Y row after row, but if necessary, it could be rewritten in order to fill X, Y column after column, as shown by X_{steps}.

4.5 Extending the Results to Block Isometric Bases

In previous sections, we showed that we can gain a factor $O(n)$ improvement when performing operations such as Gram-Schmidt decomposition on isometric bases. In this section, we show that these results can be extended to block isometric bases, that is bases that are concatenations of isometric bases.

Definition 4.12. *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_{kn}\}$ *be a basis. We say that* $\mathbf{B}$ *is block isometric if there exist* k *isometric bases* $\mathbf{B}^{(1)}, ..., \mathbf{B}^{(k)}$ *such that* $\mathbf{B}$ *is the concatenation of all these bases.*

The main idea of Algorithm 4.4 is to use the hypothesis that $r(\text{Span}(\mathbf{B}^{(i)})) = \text{Span}(\mathbf{B}^{(i)})$ (which in practice is always verified for ideal lattices) in conjunction with part 2 of Proposition 4.2: if $\tilde{\mathbf{b}}$ is the GSR of $\mathbf{b}$ w.r.t. a block $\mathbf{B}^{(i)}$, then $r(\tilde{\mathbf{b}})$ will be the GSR of $r(\mathbf{b})$ w.r.t. that same block $\mathbf{B}^{(i)}$.

Lemma 4.13. *Assume:*
- $\mathbf{B}^{(1)}, ..., \mathbf{B}^{(k)}$ *are isometric matrices for the same isometry* r*, and of same rank* n*.*
- $\forall i \in [\![1, k-1]\!], r(\text{Span}(\mathbf{B}^{(i)})) = \text{Span}(\mathbf{B}^{(i)})$.

Then Algorithm 4.4 compute the GSO of $\mathbf{B} = \{\mathbf{B}^{(1)}, ..., \mathbf{B}^{(k)}\} = \{\mathbf{b}_1, ..., \mathbf{b}_{kn}\}$ *in* $O(k^2nd)$ *elementary operations over the scalars.*

Algorithm 4.4 BlockGSO($\mathbf{B}$)

Require: Block isometric basis $\mathbf{B} = \{\mathbf{B}^{(1)}, ..., \mathbf{B}^{(k)}\} = \{\mathbf{b}_1, ..., \mathbf{b}_{kn}\}$
Ensure: Gram-Schmidt reduced basis $\tilde{\mathbf{B}}$
1: **for** $i = 0, ..., k-1$ **do**
2: $\quad\tilde{\mathbf{b}}_{ni+1} \leftarrow \mathbf{b}_{ni+1}$
3: $\quad$**for** $j = 1, ..., ni$ **do**
4: $\quad\quad\tilde{\mathbf{b}}_{ni+1} \leftarrow \mathbf{b}_{ni+1} - \frac{\langle \mathbf{b}_{ni+1}, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j$ // Make $\tilde{\mathbf{b}}_{ni+1}$ orthogonal to previous vectors
5: $\quad$**end for**
6: $\quad\tilde{\mathbf{B}}^{(i+1)} \leftarrow \{\mathbf{b}_{ni+1}, r(\mathbf{b}_{ni+1}), ..., r^{n-1}(\mathbf{b}_{ni+1})\}$
7: $\quad\tilde{\mathbf{B}}^{(i+1)} \leftarrow$ FasterIsometricGSO($\tilde{\mathbf{B}}^{(i+1)}$)
8: **end for**
9: **return** $\tilde{\mathbf{B}} = \{\tilde{\mathbf{B}}^{(1)}, ..., \tilde{\mathbf{B}}^{(k)}\}$

Proof. We prove correctness by showing inductively that at the end of each iteration i of the outer loop, the $n(i+1)$ first vectors $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_{n(i+1)}$ are the GSO of $\mathbf{b}_1, ..., \mathbf{b}_{n(i+1)}$:
- For $i = 0$, this is the case since $\tilde{\mathbf{B}}^{(1)}$ is simply the GSO of $\mathbf{B}^{(i)}$
- If it is verified until step $i-1$, then at step i the vector $\tilde{\mathbf{b}}_{ni+1}$ computed in lines 2-5 of the algorithm is exactly the GSR of $\mathbf{b}_{ni+1}$. Its rotations are orthogonal to the vectors of the previous blocks because r preserves the dot product and $\forall i, r(\text{Span}(\mathbf{B}^{(i)})) = \text{Span}(\mathbf{B}^{(i)})$, and one can verify that $\mathbf{b}_{ni+j} - r^{j-1}(\tilde{\mathbf{b}}_{ni+1}) \in \text{Span}\{\tilde{\mathbf{B}}^{(1)}, ..., \tilde{\mathbf{B}}^{(i-1)}\}$, so $r^{j-1}(\tilde{\mathbf{b}}_{ni+1})$ is exactly the orthogonalization of $\mathbf{b}_{ni+j}$ w.r.t. $\mathbf{b}_1, ..., \tilde{\mathbf{b}}_{ni}$. The basis computed at line 6 is isometric, so applying FasterIsometricGSO effectively orthogonalizes it.

We now study the complexity of algorithm 4.4. At each iteration i of the algorithm, the orthogonalization of $\mathbf{b}_{ni+1}$ w.r.t. previous vectors (steps 3 to 5) take time $O(nid)$, and steps 6-7 take time $O(nd)$. So the total complexity is $O(k^2nd)$, gaining a factor n when compared to the complexity $O((kn)^2d)$ of the naive Gram-Schmidt orthogonalization. □

The GSD can be sped up too. We will not detail it, but Fig. 4.2 gives the outline on how to use Algorithm 4.3 on a two-blocks isometric basis.

$$\tilde{\mathbf{B}} = \begin{bmatrix}\mathbf{B}^{(1)}\\ \mathbf{B}^{(2)}\end{bmatrix} \Rightarrow \begin{bmatrix}\tilde{\mathbf{B}}^{(1)}\\ \mathbf{B}^{(2)}\end{bmatrix} \Rightarrow \begin{bmatrix}\tilde{\mathbf{B}}^{(1)}\\ \{\tilde{\mathbf{b}}_{n+1}, ..., r^{n-1}(\tilde{\mathbf{b}}_{n+1})\}\end{bmatrix} \Rightarrow \begin{bmatrix}\tilde{\mathbf{B}}^{(1)}\\ \tilde{\mathbf{B}}^{(2)}\end{bmatrix}$$

$$\mathbf{L} = \left[\begin{array}{c|c} I_n & 0_n \\ \hline 0_n & I_n \end{array}\right] \Rightarrow \left[\begin{array}{c|c} \mathbf{L}_1 & 0_n \\ \hline 0_n & I_n \end{array}\right] \Rightarrow \left[\begin{array}{c|c} \mathbf{L}_1 & 0_n \\ \hline \mathbf{L}_3 & I_n \end{array}\right] \Rightarrow \left[\begin{array}{c|c} \mathbf{L}_1 & 0_n \\ \hline \mathbf{L}_3 & \mathbf{L}_4 \end{array}\right]$$

Figure 4.2: Computing the GSD of a two-block isometric basis. $\tilde{\mathbf{B}}$ and $\mathbf{L}$ always satisfy $\mathbf{L} \times \tilde{\mathbf{B}} = \mathbf{B}$

4.6 GSO and GSD in Exact Arithmetic

Generally, GSO and GSD are performed over real bases, so the standard way of implementing it is by using floating-point arithmetic. However, this can result in rounding errors: several books and articles discuss this problem with a good introduction being [Hig02].

When the input vectors are in $\mathbb{Z}^d$, as it is very often the case in lattice-based cryptography, then the GSD can be performed using only exact arithmetic over $\mathbb{Q}$. Moreover, some algorithms such as the original LLL algorithm [LLL82] explicitly perform exact GSD.

However, this gain in precision comes at the cost of reduced efficiency: when an integer basis undergoes GSO, the reduced vectors' bitsize quickly escalates in the dimension of the basis and of the underlying space. This phenomenon is called *coefficient explosion* and impacts the space *and* computational cost of GSD. In this section, we adapt Algorithms 4.2 and 4.3 to the exact arithmetic setting and show that we still gain a $O(d)$ factor compared to classical GSO/GSD. Moreover, our adapted algorithms completely avoid rational arithmetic.

Through this section, we make an additional "niceness" assumption over the isometry r, namely we suppose that it maps integer vectors into integer vectors: $\forall \mathbf{b} \in \mathbb{Z}^d, r(\mathbf{b}) \in \mathbb{Z}^d$.

4.6.1 GSO in Exact Arithmetic

Definition 4.14. *Let $\mathbf{B} = (\mathbf{b}_j)_{1 \leqslant j \leqslant n}$ be an isometric basis, and for $j \in [\![1, n]\!]$, $\tilde{\mathbf{b}}_j, \mathbf{v}_j, C_j, D_j$ be defined as in Section 4.3. We then define, $\forall i, j \in [\![1, n]\!]$, the following values:*

- $\lambda_{j,j} = \prod_{1 \leqslant k \leqslant j} \|\tilde{\mathbf{b}}_k\|^2$
- $\tilde{\mathbf{db}}_j = \lambda_{j-1,j-1}\tilde{\mathbf{b}}_j$
- $c_j = \lambda_{j-1,j-1}C_j$
- $\lambda_{i,j} = L_{i,j}\lambda_{j,j}$
- $\mathbf{dv}_j = \lambda_{j-1,j-1}\mathbf{v}_j$
- $d_j = \lambda_{j-1,j-1}D_j$

Proposition 4.15. *Using notations of Definition 4.14, $\forall i, j \in [\![1, n]\!]$, we have:*

1. $\lambda_{i,j} \in \mathbb{Z}$
2. $\tilde{\mathbf{db}}_j, \mathbf{dv}_j \in \mathbb{Z}^d$
3. $c_j, d_j \in \mathbb{Z}$

Proof. Proofs for assertions 1 and 2 can be found per example in [Gal12, chapter 17, theorem 17.3.2]. As for assertion 3, $d_j = \lambda_{j,j}$ and $c_j = \langle \mathbf{v}_1, r(\tilde{\mathbf{db}}_j) \rangle$, where $\mathbf{v}_1$ and $r(\tilde{\mathbf{db}}_j)$ are in $\mathbb{Z}^d$. □

With these results in hand, we can now devise an integer version of Algorithm 4.2. Instead of outputting rational values, Algorithm 4.5 outputs only integers and integer

vectors, and one can then retrieve any vector $\tilde{\mathbf{b}}_k$ by computing $\tilde{\mathbf{b}}_k = \frac{1}{\sqrt{d_{k-1}}}\tilde{\mathbf{db}}_k$. Algorithm 4.5 uses no rational number and all the internal operations, including exact divisions in steps 6,7 and 9, output integer values. The following lemma shows that in the case we use exact arithmetic, Algorithm 4.5 is still at least $O(n)$ faster than standard GSO.

Algorithm 4.5 IntegerIsometricGSO($\mathbf{B}$)

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$
Ensure: $(\tilde{\mathbf{db}}_k, \mathbf{dv}_k, c_k, d_k)_{k=1...n}$ as defined in Definition 4.14
1: $\tilde{\mathbf{db}}_1 \leftarrow \mathbf{b}_1$
2: $\mathbf{dv}_1 \leftarrow \mathbf{b}_1$
3: $c_1 \leftarrow \langle r(\tilde{\mathbf{b}}_1), \mathbf{dv}_1 \rangle$
4: $d_0 \leftarrow 1, d_1 \leftarrow \|\mathbf{b}_1\|^2$
5: **for** $k = 1, ..., n-1$ **do**
6: $\quad \tilde{\mathbf{db}}_{k+1} \leftarrow \left[d_k r(\tilde{\mathbf{db}}_k) - c_k \mathbf{dv}_k\right] / d_{k-1}$
7: $\quad \mathbf{dv}_{k+1} \leftarrow \left[d_k \mathbf{dv}_k - c_k r(\tilde{\mathbf{db}}_k)\right] / d_{k-1}$
8: $\quad c_{k+1} \leftarrow \langle \mathbf{v}_1, r(\tilde{\mathbf{db}}_{k+1}) \rangle$
9: $\quad d_{k+1} \leftarrow \frac{d_k^2 - c_k^2}{d_{k-1}}$
10: **end for**
11: **return** $(\tilde{\mathbf{db}}_k, \mathbf{dv}_k, c_k, d_k)_{k=1...n}$

Lemma 4.16. *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in (\mathbb{Z}^d)^n$ *be an integral isometric basis,* $|\mathbf{B}| = \max_{k=1...n}(\|\mathbf{b}_k\|)$ *and* $\mathcal{M}(X)$ *denote the time complexity for multiplying two integers of at most* X *bits. Suppose the isometry* r *associated to* $\mathbf{B}$ *can be computed in time and space linear to the size of the input. Then Algorithm 4.5 performs in time* $O(dn\mathcal{M}(n \log |\mathbf{B}|))$.

Proof. By definition, $d_k = \prod_{1 \leqslant i \leqslant k} \|\tilde{\mathbf{b}}_i\|^2$ so $|d_k| \leqslant |\mathbf{B}|^{2k}$. Moreover, $|C_k| < D_k$ implies $|c_k| < d_k$ and therefore c_k, d_k both have bitsizes $O(k \log |\mathbf{B}|)$. On the other hand, $\tilde{\mathbf{db}}_k$ (resp. $\mathbf{dv}_k$) has its norm less than $|\mathbf{B}|^{2k-1}$ so the four scalar-vectors products performed on steps 6,7 have complexity $O(d\mathcal{M}(k \log |\mathbf{B}|))$, as well as the two divisions of vectors by scalars (we recall that euclidean division of X bit numbers can be performed in time $O(\mathcal{M}(X))$). Overall, each iteration k of the **for** loop takes time $O(d\mathcal{M}(k \log |\mathbf{B}|))$, so the total complexity of Algorithm 4.5 is $O(dn\mathcal{M}(n \log |\mathbf{B}|))$.

□

4.6.2 GSD in Exact Arithmetic

The isometric GSD can also naturally be converted into an efficient, "rational-free" version. Let $x_{i,j} \triangleq \lambda_{j-1} X_{i,j} = \langle \mathbf{b}_i, \tilde{\mathbf{db}}_j \rangle \in \mathbb{Z}$ and $y_{i,j} \triangleq \lambda_{j-1} Y_{i,j} = \langle r(\mathbf{b}_i), \mathbf{dv}_j \rangle \in \mathbb{Z}$. The relations

$$\begin{cases} X_{i+1,j+1} = X_{i,j} - \frac{C_j}{D_j} Y_{i,j} \\ Y_{i,j+1} = Y_{i,j} - \frac{C_j}{D_j} X_{i,j} \end{cases}$$

then become

$$\begin{cases} x_{i+1,j+1} = \frac{d_j x_{i,j} - c_j y_{i,j}}{d_{j-1}} \\ y_{i,j+1} = \frac{d_j y_{i,j} - c_j x_{i,j}}{d_{j-1}} \end{cases}$$

The $x_{i,j}$'s actually are the $\lambda_{i,j}$'s, but in Algorithm 4.6 we continue to write $x_{i,j}$ since it highlights the natural transformation of Algorithm 4.3 to Algorithm 4.6.

Algorithm 4.6 IntegerIsometricGSD($\mathbf{B}, (c_k, d_k)_{k=1...n}$)

Require: Basis $\mathbf{B}$ and the values $(c_k, d_k)_{k=1...n}$
Ensure: $x_{i,j}$'s,$y_{i,j}$'s as defined above
1: **for** $i = 2...n$ **do** // Computing the $(x_{i,1}), (y_{i,1})$
2: $x_{i,1} \leftarrow \langle \mathbf{b}_i, \tilde{\mathbf{b}}_1 \rangle$
3: $y_{i,1} \leftarrow \langle r(\mathbf{b}_i), \mathbf{b}_1 \rangle$
4: **for** $j = 2...i-1$ **do**
5: $x_{i,j} \leftarrow \left[d_{j-1} x_{i-1,j-1} - c_{j-1} y_{i-1,j-1} \right] / d_{j-2}$
6: $y_{i,j} \leftarrow \left[d_{j-1} y_{i,j-1} - c_{j-1} x_{i,j-1} \right] / d_{j-2}$
7: **end for**
8: **end for**
9: **return** $(x_{i,j}, y_{i,j})_{1 \leqslant i,j \leqslant n}$

Lemma 4.17. *Following the notations of Lemma 4.16, the time complexity of Algorithm 4.6 is* $O(n^2 \mathcal{M}(n \log |\mathbf{B}|) + nd\mathcal{M}(\log |\mathbf{B}|))$.

Proof. The costliest operations of Algorithm 4.6 are either the $2(n-1)$ dot products in steps 2 and 3, which cost $O(d\mathcal{M}(\log |\mathbf{B}|))$ each, or the essentially $3n^2$ multiplications and divisions made at steps 5 and 6, which cost $O(\mathcal{M}(j \log |\mathbf{B}|))$ each. Summing these costs yields the result. □

Note that in practice d is not much bigger than n, so the complexity of Algorithm 4.6 becomes $O(n^2 \mathcal{M}(n \log |\mathbf{B}|))$. Even in exact arithmetic, our GSO and GSD algorithms still perform $O(n)$ times faster than standard GSD, which complexity is $O(dn^2 \mathcal{M}(n \log |\mathbf{B}|))$ (implicit in the proof of [Gal12, Theorem 17.3.4]). Moreover, we manage to avoid the use of any rational number, making our algorithms both efficient and easy to implement.

4.7 NTRU Lattices

NTRU lattices are a special class of lattices widely used in cryptography, because their ring structure allows a gain of a factor n both in time and space when performing usual operations over lattices. This results in efficient and compact cryptosystems (e.g. [HPS98, LTV12, DDLL13]).

Let $N, q \in \mathbb{N}^*$ and $f, g, F, G \in \mathbb{Z}[x]/(x^N + 1)$ such that $fG - gF = q \bmod (x^N + 1)$. The NTRU lattice generated by f, g, F, G is the lattice generated by the rows of the block matrix

$$\left[\begin{array}{c|c} \mathcal{A}(f) & \mathcal{A}(g) \\ \hline \mathcal{A}(F) & \mathcal{A}(G) \end{array}\right]$$

Where $\mathcal{A}(p) = \mathcal{A}_{\phi_{2N}}(p)$ is the $N \times N$ matrix which i-th row is the coefficients of $x^{i-1} \cdot p(x) \bmod (x^N + 1)$.

In [GHN06], Gama et al. considered exact GSD of NTRU bases. They showed that these lattices verify an algebraic property called symplecticity, which allows them to compute the exact GSD faster than with the standard algorithm, using [GHN06, Cor. 1].

But in addition to being q-symplectic, NTRU bases are also block isometric. So we devised an algorithm to compute the exact GSD of a NTRU basis, by combining three strategies:

- use Algorithms 4.5 and 4.6 in order to avoid rational arithmetic, as in [Gal12, GHN06].

Table 4.1: Timings for Gram-Schmidt over NTRU bases, in seconds. The implementation was done on `Sage 5.3`. Timings were performed on an Intel Core i5-3210M laptop with a 2.5GHz CPU and 6GB RAM. Isometric GSD is "standard" GSD for block isometric bases, whereas Iso.+Symp. GSD takes into account the observations from [GHN06].

Dimension $n = 2N$	128	256	512	1024
Standard GS [GHN06]	3.22	30.7	390	4536
Dual GS [GHN06]	2.39	17	214	2496
Symplectic GS [GHN06]	0.89	5.73	33.9	279
Isometric GSD	0.48	2.05	12.4	89
Iso.+Symp. GSD	0.312	1.4	8.18	57.8

- use the GSO/GSD strategies for isometric bases detailed in Section 4.5.
- use [GHN06, Corollary 1] to compute one half of the GSO and get the other for free.

We compared our exact reduction algorithm with the ones from [GHN06]. It turns out that our algorithm is faster, both theoretically and in practice, despite computing more information: it provides $\tilde{\mathbf{B}}$ and $\mathbf{L}$, whereas the algorithms in [GHN06] only provide $\mathbf{L}$. The timings are summarized in Table 4.1 and the full implementation can be found at:

`https://github.com/tprest/Fast-GSD`

4.8 Reversibility and Application to Linear-Storage Klein Sampling

A drawback of applying Klein's sampler (Section 2.4.1, Algorithm 2.6) over ring lattices is that, even though the basis $\mathbf{B}$ of an ring lattice can be stored using $O(1)$ vectors, this is not the case for the reduced basis $\tilde{\mathbf{B}}$, which needs n vectors. This can quickly impede the practicality of Klein's sampler: for example, for $n = 1024$ (a typical dimension for cryptographic lattices), if $\tilde{\mathbf{B}}$ is stored using 128 bits of precision, the bitsize of $\tilde{\mathbf{B}}$ then exceeds 128 Mbits.

The Levinson recursion allows to overcome this problem by computing the reduced basis $\tilde{\mathbf{B}}$ on-the-fly. An obstacle is that Klein's sampler needs the vectors of $\tilde{\mathbf{B}}$ *in reverse order*, so a straightforward use of Algorithm 4.1 or 4.2 does not solve the problem since it provides the basis in direct order. Fortunately, as Figure 4.1 suggests, we can reverse the order in which the Levinson recursion performs its operations, as illustrated in Figure 4.3. More importantly, this can be done in linear space, a property that Klein's sampler can take advantage of to be itself executed in linear space instead of quadratic.

Definition 4.18. *For a basis* $\mathbf{B}$*, we denote, for any* $i \in [\![1; n-1]\!]$*,* $C_i = \langle \mathbf{v}_i, r(\tilde{\mathbf{b}}_i) \rangle$ *and* $D_i = \|\tilde{\mathbf{b}}_i\|^2$*. We also define* $H_i = \frac{1}{1-(C_i/D_i)^2} = \frac{D_i}{D_{i+1}}$ *and* $I_i = \frac{C_i/D_i}{1-(C_i/D_i)^2} = \frac{C_i}{D_{i+1}}$.

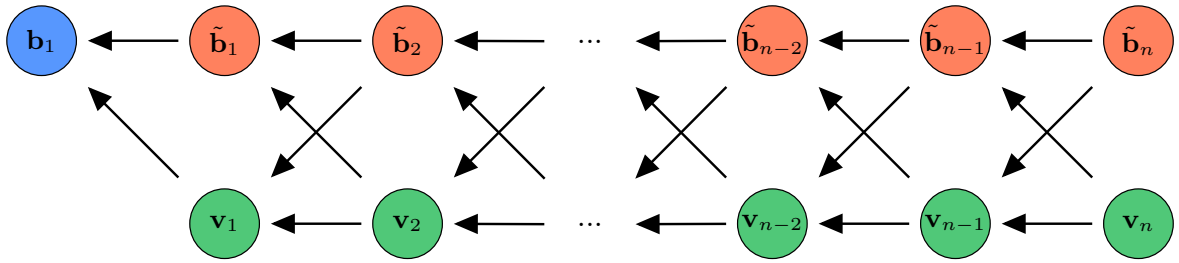


Figure 4.3: Reversing the execution of Algorithm 4.1

Algorithm 4.7 CompactKlein$(\mathbf{B}, \sigma, \mathbf{c}, \tilde{\mathbf{b}}_n, \mathbf{v}_n, (H_i, I_i)_i)$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$, center $\mathbf{c} \in \mathbb{Z}^m$, precomputed vectors $\tilde{\mathbf{b}}_n, \mathbf{v}_n$, precomputed values $(H_i, I_i)_{1 \leqslant i < n}$ from definition 4.18
Ensure: $\mathbf{z}$ sampled in $D_{\mathcal{L}(\mathbf{B}),\sigma,\mathbf{c}}$

1: $\mathbf{c}_n \leftarrow \mathbf{c}$
2: **for** $i \leftarrow n, ..., 1$ **do**
3: $\quad d_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$
4: $\quad \sigma_i \leftarrow \sigma / \|\tilde{\mathbf{b}}_i\|$
5: $\quad z_i \leftarrow \lfloor d_i \rceil_{\sigma_i}$
6: $\quad \mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \mathbf{b}_i$
7: $\quad \tilde{\mathbf{b}}_{i-1} \leftarrow r^{-1}(H_{i-1}\tilde{\mathbf{b}}_i + I_{i-1}\mathbf{v}_i)$
8: $\quad \mathbf{v}_{i-1} \leftarrow I_{i-1}\tilde{\mathbf{b}}_i + H_{i-1}\mathbf{v}_i$
9: **end for**
10: **return** $\mathbf{c} - \mathbf{c}_0$

Lemma 4.19. *Algorithms 2.6 and 4.7 produce the same output when they have the same input* $\mathbf{B}$ *and* $\mathbf{c}$ *(assuming the associated precomputed values are correct).*

Proof. First, observe that $\forall i \in [\![1, n-1]\!]$, $|C_i| < D_i$, because otherwise D_{i+1} would be zero and $\mathbf{B}$ would not be a basis. One can see that the "linear system"

- $\tilde{\mathbf{b}}_{i+1} = r(\tilde{\mathbf{b}}_i) - \frac{C_i}{D_i}\mathbf{v}_i$
- $\mathbf{v}_{i+1} = \mathbf{v}_i - \frac{C_i}{D_i} r(\tilde{\mathbf{b}}_i)$

is invertible :

- $\tilde{\mathbf{b}}_i = r^{-1}(H_i\tilde{\mathbf{b}}_{i+1} + I_i\mathbf{v}_{i+1})$
- $\mathbf{v}_i = I_i\tilde{\mathbf{b}}_{i+1} + H_i\mathbf{v}_{i+1}$

H_i and I_i are always defined since $|C_i| < D_i$. Therefore, the same way the values C_i, D_i allow to compute $\tilde{\mathbf{b}}_{i+1}, \mathbf{v}_{i+1}$ from $\tilde{\mathbf{b}}_i, \mathbf{v}_i$, H_i, I_i allow to compute $\tilde{\mathbf{b}}_i, \mathbf{v}_i$ from $\tilde{\mathbf{b}}_{i+1}, \mathbf{v}_{i+1}$. □

This allows us to use Klein's sampler using $O(m)$ memory space instead of $O(mn)$ for the classic version. The overhead in time is reasonable:

- Klein's sampler: $2mn$ additions, $2mn$ multiplications, n samplings in $\mathbb{Z}$
- Compact Klein's sampler: $4mn$ additions , $6mn$ multiplications, n samplings in $\mathbb{Z}$

Therefore, the compact Klein sampler is *at most* three times slower than the classic one. This is confirmed by experiments summarized in Table 4.2. Moreover, in Algorithm 4.7, it is possible to sample around several $\mathbf{c}$'s at the same time: this then makes negligible the overhead induced by the addition (in Algorithm 4.7) of lines 7 and 8. This time-memory

trade-off allows to use Klein's sampler for k targets in space $O(km)$ and in time *at most* $\left(1+\frac{2}{k}\right)$ times the time required by the classic Klein sampler.

4.8.1 Precision Analysis of the Compact Klein's Sampler

In Section 3.4.2, we gave upper bounds on the precision required for Klein's sampler to be indistinguishable from a sampler with infinite precision. For the compact Klein's sampler, an additional variable has to be taken into account: the vectors of the orthogonalized basis $\tilde{\mathbf{B}}$ are not known in advance, but computed themselves on-the-fly, and therefore, they may be subject to error propagation themselves. A precision analysis of the on-the-fly computation of $\tilde{\mathbf{B}}$ is therefore needed.

A standard approach would be to compute worst-case bounds on the precision with which the inputs $(\tilde{\mathbf{b}}_n, \mathbf{v}_n, (H_i, I_i)_i)$ of Algorithm 4.7 must be known in order for the precision of the computed matrix $\tilde{\mathbf{B}}$ to fall in the bounds of Lemma 3.12. However, this approach is inefficient here because the precision required in practice will be much less than in theory due to the fact that theoretically, there could exist bases on which our "computing $\tilde{\mathbf{B}}$ on-the-fly" procedure is rather unstable.

Instead, we favor a different approach: for a given basis $\mathbf{B}$, we compute the associated values $(\tilde{\mathbf{b}}_n, \mathbf{v}_n, (H_i, I_i)_i)$ at very high precision, and then estimate how many bits of precision for $(\tilde{\mathbf{b}}_n, \mathbf{v}_n, (H_i, I_i)_i)$ are required in order that $\tilde{\mathbf{B}}$ is in the bounds of Lemma 3.12 This can be done as a precomputation, and once we do it, we know for sure that every time we run Algorithm 4.7, the matrix $\tilde{\mathbf{B}}$ will be in the bounds of Lemma 3.12, because computing it is a deterministic process which does not depend on the center $\mathbf{c}$ but only on $(\tilde{\mathbf{b}}_n, \mathbf{v}_n, (H_i, I_i)_i)$.

Our practical experiments strongly suggest that for each coordinate of $\tilde{\mathbf{B}}$, no more than 30 bits of precision are lost in the case of NTRU lattices of dimension $2N \leqslant 4096$.

4.8.2 Space Complexity of the Compact Klein's Sampler

Suppose that $\tilde{\mathbf{B}}$ needs to be known up to $|\log_2 \delta|$ bits, for some $\delta < 1$. In order to be able to run Algorithm 4.7 any time (without having to undergo the GSO beforehand), one only needs to store $(H_i, I_i, \|\tilde{\mathbf{b}}_i\|)_{i=1\ldots n}$ as well as $\tilde{\mathbf{b}}_n, \mathbf{v}_n$. However, it is straightforward to use the relation $\frac{\|\tilde{\mathbf{b}}_{i+1}\|^2}{\|\tilde{\mathbf{b}}_i\|^2} = 1 - \left(\frac{C_i}{D_i}\right)^2$ to save even more space by just storing the $\|\tilde{\mathbf{b}}_i\|$'s and deriving the H_i's, I_i's from them.

During the execution of Algorithm 4.7, one also needs to store the current $\tilde{\mathbf{b}}_i, \mathbf{v}_i$. So overall the space requirement of Algorithm 4.7 is $5n(|\log_2 \delta| + b)$, where b is the "number of bits lost" in steps 6, 7 of Algorithm 4.7: in other words, b is such that if $\tilde{\mathbf{b}}_n, \mathbf{v}_n, (\|\tilde{\mathbf{b}}_i\|)_{i=1\ldots n}$ are known up to $|\log_2 \delta| + b$ bits, then $\tilde{\mathbf{B}}$ is guaranteed to be known up to $|\log_2 \delta|$ bits.

For NTRU lattices, this analysis can be refined: only half of the $\|\tilde{\mathbf{b}}_i\|$ need to be known, and $\tilde{\mathbf{b}}_n, \mathbf{v}_n$ can be determined from $\mathbf{b}_1 = \tilde{\mathbf{b}}_1$ [GHN06, Corollary 1]. Instead of needing to know $3n(|\log_2 \delta| + b)$ bits beforehand, we just need $\frac{n}{2}(|\log_2 \delta| + b)$, so the total space requirement is $2.5n(|\log_2 \delta| + b)$.

4.9 Concrete Space Requirement of the Compact Klein's Sampler

In the previous sections, we showed that Klein's sampler could be used in conjunction with a "reversed" execution of the Levinson recursion to save space by a factor $O(n)$. However, to quantify in practice how much space we can save, it is important to know at which precision this data needs to be stored.

Indeed, that for σ big enough, the output f of Algorithm 2.6 is statistically close to the distribution $\mathcal{D}_{\mathbf{L}(\mathbf{B}),\sigma,\mathbf{c}}$. However, the proof holds only when $\mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c}$ and the values $\|\tilde{\mathbf{b}}_i\|$'s are known exactly. But in practice, one can not afford to do computations with the exact representation of $\tilde{\mathbf{B}}$ and of the $\|\tilde{\mathbf{b}}_i\|$'s, as it would be too costly in terms of space and computational resources. Therefore, $\tilde{\mathbf{B}}$ and the $\|\tilde{\mathbf{b}}_i\|$'s are stored up to some finite precision, and this finite precision introduces errors which impact the output distribution of the algorithm.

In a previous version of this work [LP15, Sections 9 and A], this was done by computing directly how imprecisions in knowing $\tilde{\mathbf{B}}$ and the $\|\tilde{\mathbf{b}}_i\|$ affected the statistical distance between the output of Klein's sampler with exact input (that we will denote $\mathcal{P}_0$) or input imprecise up to an uncertainty δ for each coordinate of $\tilde{\mathbf{B}}$ (we denote the output distribution $\mathcal{P}_\delta$). These computations were arguably tedious to read, so we use instead the results from Section 3.4.2, which gives bounds on the KL divergence between $\mathcal{P}_0$ and $\mathcal{P}_\delta$. Getting a bound on the statistical distance from the computations in Section 3.4.2 is straightforward, so we do not detail them here.

In most applications, we could probably stick with the KL divergence as it allows to roughly halve to required precision, but it gives slightly less generic security arguments than statistical distance, so we chose to take the latter as our security metric. Besides, choosing one metric or the other doesn't affect significantly the ratio between the storage requirements of our compact sampler and the classic one.

As an example, we took a NTRU basis for $N = 1024, q = 1024$. This allows some simplifications to the formula of Lemma 3.12: $m = n$ and we can assume that $\|\tilde{\mathbf{b}}_i\| \approx 1.17\sqrt{q}$ (this is supported by facts in Chapter 6). This yields precisions that are given in Table 4.2.

Table 4.2: Timings (in milliseconds) and space requirements (in bits and mega-bits) of the classic and compact Gaussian Samplers (Classic GS and Compact GS). The implementation was done in `C++` using `GMP`. Timings were performed on an Intel Core i5-3210M laptop with a 2.5GHz CPU and 6GB RAM.

Statistical distance from ideal		2^{-80}	2^{-128}	2^{-192}
Precision needed	Classic GS	115 bits	163 bits	223 bits
	Compact GS	145 bits	193 bits	253 bits
Running time	Classic GS	115 ms	170 ms	203 ms
	Compact GS	446 ms	521 ms	523 ms
Space requirements	Classic GS	115 Mb	163 Mb	223 Mb
	Compact GS	0.35 Mb	0.47 Mb	0.63 Mb

As a test for the practicality of our compact Gaussian Sampler, we implemented both the classic and compact Gaussian Samplers and compared their timings and space

requirements. As predicted by our computations, the compact Gaussian Sampler is not much more than thrice slower (the slight precision growth makes it a bit slower than predicted) than the classic one, while having space requirements smaller by between two and three orders of magnitude. Our results are summarized in Table 4.2 and the complete implementation can be found on https://github.com/tprest/Compact-Sampler.

4.10 Conclusion

In this work, we showed that the vast majority of bases used in *efficient* lattice-based cryptography are compatible with an algorithm known as the Levinson recursion which allows to orthogonalize them very efficiently. This led to orthogonalization of NTRU faster than the prior art [GHN06], as well as reducing the quasiquadratic space complexity of Klein's sampler by a linear factor.

If would be interesting to see if the Krylov structure (see Section 4.1.3) that underlies these bases can be exploited in other ways. Recent cryptanalytic works on these specific bases have more focused on the algebraic number theoretic part [Ber14, CDPR15], but this chapter has shown that exploiting "simple" algebraic and geometric properties can be very efficient too.

Part II

New Gaussian Samplers over Ring Lattices

After improving the existing Gaussian samplers, our goal is to build new ones. Many of the most efficient cryptographic constructions are instantiated over ring lattices, which are $\mathcal{R}$-modules over some ring $\mathcal{R}$. In this part of the thesis, we devise new Gaussian samplers taking actively advantage of this structure over $\mathcal{R}$ and compare them to usual samplers. This is done in three strongly interdependent chapters.

The Chapter 5 is our first take at exploiting the ring structure that underlies many families of cryptographic lattices. In this chapter, we consider rings $\mathcal{R}$ that are the rings of integers $\mathcal{O}_{\mathbb{K}}$ of a number field $\mathbb{K}$ – these include $\mathcal{R} = \mathbb{Z}[x]/(x^n - 1)$ and cyclotomic rings $\mathcal{R} = \mathbb{Z}[x]/(\phi_m(x))$, both of which are widespread in ring lattice-based cryptography. Our sampler works for lattices over $\mathcal{R}$, instead of usual lattices over $\mathbb{Z}$. At a high level this algorithm works like Klein's sampler, but as a subroutine calls an instantiation of Peikert's sampler over $\mathcal{R}$. The resulting *Hybrid sampler* is a quality-speed trade-off between Klein's and Peikert's samplers. But as it is crafted specifically to take advantage of the ring structure over $\mathcal{R}$ (unlike Peikert's sampler which "passively" benefits from this structure), one may hope that in practice it takes the best of each sampler. We verify this assumption on a practical example in the next chapter.

In Chapter 6, we take the Full-Domain Hash signature scheme defined in [GPV08] and instantiate it over NTRU lattices, a compact class of lattices that allow efficient instantiations [DDLL13, DLP14]. As this signature scheme uses Gaussian sampling as a core procedure, taking either Klein's, Peikert's or the Hybrid sampler will yield different trade-offs between speed and security (assuming the dimension and modulus of the lattice are fixed). After conducting extensive experiments and heuristics, it turns out that the Hybrid sampler is close to taking the best of both worlds: it is about twice slower than Peikert's sampler and $\tilde{O}(n)$ times faster than Klein's sampler, but using it yields a scheme whose security is closer to Klein's. Additionally, the scheme we devised yields signatures among the most compact in lattice-based cryptography, though its speed still needs to be evaluated in practice.

In Chapter 7, we discover that the ideas of the fast Fourier transform can be applied to speed up the orthogonalization process of a circulant matrix. When d is composite, we show that it is possible to proceed to the orthogonalization in an inductive way, leading to a structured Gram-Schmidt decomposition. In turn, this structured Gram-Schmidt decomposition accelerates the nearest plane algorithm, and makes it asymptotically as fast as the round-off algorithm on circulant matrices. The results easily extend to *cyclotomic rings*, and can be easily adapted to Gaussian Samplers.

A small part of Chapter 6 was published in ***Efficient Identity-Based Encryption over NTRU Lattices*** [DLP14], a joint paper with Léo Ducas and Vadim Lyubashevsky which was published at ASIACRYPT. All the rest is joint work with Léo Ducas.

Chapter 5

A Hybrid Gaussian Sampler for Ring Lattices

5.1 Introduction

Currently, two main algorithms allow to do Gaussian Sampling over arbitrary lattices. The first one is a randomized version of Babai's nearest-plane algorithm [Bab85, Bab86] due to Klein [Kle00] and Gentry, Peikert, Vaikuntanathan [GPV08]. It is also the one that produces the shortest vectors but it has a sequential structure and runs in time $\tilde{O}(n^2)$, even on ring lattices.

The second one is due to Peikert [Pei10] and can be seen as a randomized version of Babai's rounding algorithm [Bab85, Bab86]. In essence, Peikert's sampler is very different from Klein's as its underlying idea essentially is that the convolution of two Gaussians is also a Gaussian. In practice, it performs very differently from Klein's sampler as it is parallelizable. If the underlying lattice is endowed with a ring structure, Peikert's sampler can take advantage of it and run in quasilinear time $\tilde{O}(n)$, where n is the dimension of the lattice. However, its quality is worse than Klein's sampler—that is, the size of the outputted vector is significantly longer—which dilutes the security of underlying cryptographic constructions.

5.1.1 Our Contribution

The main contribution of this chapter is a new discrete Gaussian sampler over ring lattices. Our algorithm is constructed in two steps. First, we give ring variants for Klein's and Peikert's samplers. Given a ring $\mathcal{R}$, the ring variant of Klein's sampler is a generalization from sampling in lattices in $\mathbb{Z}^m$ to lattices in $\mathcal{R}^m$, whereas the ring variant of Peikert's sampler is simply an instantiation over $\mathcal{R}$.

The ring variant of Klein's sampler may run faster over ring bases than the original algorithm. However, in order to do so it needs to invoke a fast sampler over $\mathcal{R}$ as an internal oracle. So we use the ring variant of Peikert's sampler to be this internal oracle. The resulting algorithm is a hybrid sampler: at high-level it operates as Klein's sampler, but at low level it uses Peikert's algorithm. Our hybrid sampler allows a trade-off between the slow but high-quality sampler of Klein, and the fast but lower-quality sampler of Peikert. As a by-product, we also obtain a hybrid between Babai's nearest plane and rounding algorithms.

The practical value of our hybrid sampler will then be assessed in the next chapter, which also provides a practical signature scheme over NTRU lattices based on the Full-Domain Hash signature framework of [GPV08].

5.1.2 Motivation

To understand the efficiency gain that lattices over rings allows, it is important to understand the relation that this structure brings between vectors, matrices and polynomials, which can be summarized as a "triple duality".

A bit more formal explanation for the advantages provided by this ring structure is that for any cyclotomic polynomial ϕ_m, there exists obvious isomorphisms between $\mathcal{A}_{\phi_m}(\mathbb{R}[x]), \mathbb{R}^{\varphi(m)}$ and $\mathbb{R}[x]/(\phi_m(x))$. If we work with matrices and vectors in $\mathcal{A}_{\phi_m}(\mathbb{R}[x]), \mathbb{R}^n$ (where $n = \varphi(m)$), we can therefore gain a factor at least $\tilde{O}(n)$ when performing some operations (matrix addition, product, inversion, determinant and matrix-vector product to cite the most common).

This is the fundamental reason that leads many ring lattice-based schemes to be very efficient. What motivated this work was to devise a Gaussian sampler that would actively take advantage of the ring structure. More explicitly, given a ring $\mathcal{R}$, we wanted to craft a sampler that would both be time-efficient and sample good quality vectors over $\mathcal{R}$-lattices.

5.1.3 Choice and Implementation of the Ring

In essence, our techniques are independent of the choice of the number field and its ring of integers, and they are relevant as long as the chosen field admits a fast multiplication algorithm.

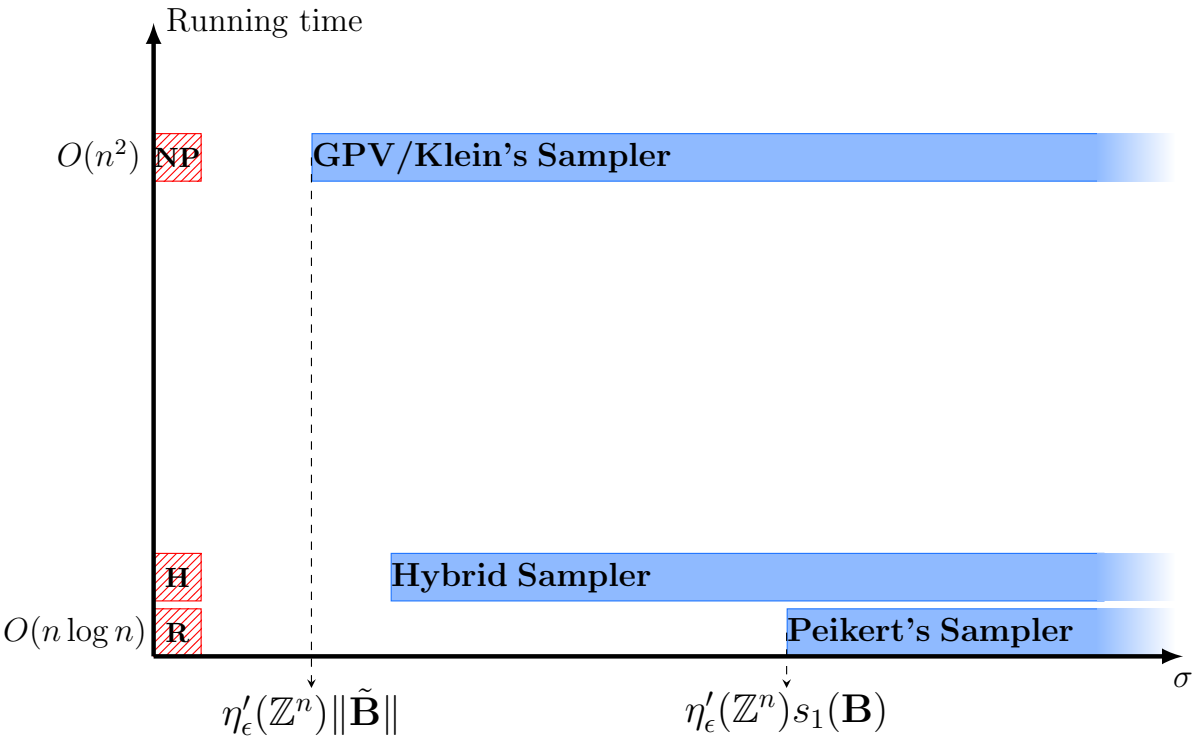
Our results are tested with typical rings used in lattice-based cryptography so far, that is the ring of integers of the m-th cyclotomic field for m a power of 2. Other choices are of course possible, but choosing a ring comes with the task of studying its geometry (that is essentially finding a good $\mathbb{Z}$-basis of that ring). The cyclotomic cases have been treated extensively [LPR10, DD12, LPR13b].

In the light of recent cryptanalytic developments [Ber14, CDPR15], it is worth noting that our result also applies to the so-called NTRU-prime rings $\mathcal{R} = \mathbb{Z}[x]/(x^p - x - 1)$ as proposed by Bernstein [Ber14], yet the geometric aspect of such rings remains to be studied.

5.1.4 Related Work

The seminal work of [GPV08] spawned a lot of papers trying to improve it. The most notable may be [Pei10], which created a completely different sampler.

An important contribution is the work of [MP12], which introduces the use of a public matrix $\mathbf{G}$ to generate a random matrix $\mathbf{A}$ along with some trapdoor information allowing to do very efficient Gaussian sampling on $\mathbf{L}_q^\perp(\mathbf{A})$. However, to the best of our knowledge these techniques do not apply to specific families of lattices such as NTRU lattices. More importantly, these techniques imply a blowup in the parameters of the lattice: the public key is a matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$, where n is rank of the lattice and $m > n \log q$. In comparison, NTRU lattice only have $m = 2n$. It will however be interesting in future work to see if the parameters of [MP12] can be relaxed to obtain lattices that approach NTRU lattices in compactness.



Our contributions and their performances compared to existing algorithms. On the left are Babai's algorithms: nearest plane (NP), hybrid (H) and rounding (R). On the right are the Gaussian Samplers. For samplers, the lower σ can be set, the better.

Figure 5.1: Complexity and Quality Comparison of the Gaussian Samplers

Floating point arithmetic issues were partly addressed in [DN12a], which speeds up both Klein's and Peikert's samplers to $\tilde{O}(n^2)$. For lattices over rings, [DN12a] also uses the ring structure of the lattice to get Peikert's offline phase to reach time and space complexity $\tilde{O}(n)$.

Rejection sampling techniques are used in [BLP+13] in order to use Klein's sampler with an even shorter standard deviation. The geometric properties of lattices over rings are used in [LP15] to reduce the space requirement of Klein's sampler to $\tilde{O}(n)$.

5.1.5 Roadmap

In Section 5.2, we set up the definitions and notations that we will use throughout the chapter. In Section 5.3, we define the ring variants of Klein's and Peikert's samplers. In Section 5.4, we introduce our hybrid sampler and prove its correctness. In the ulterior Chapter 6, we assess the practical interest of our new sampler by comparing its efficiency and time complexity to those of Klein's and Peikert's algorithms.

5.2 Preliminaries

5.2.1 Notations

We recall that most of the notation conventions are stated in pages 11 to 19. Vectors will be written with lowercase bold letters, matrices and bases in uppercase bold letters, and

scalars (which includes ring and field elements) in non-bold letters. A matrix $\mathbf{B} \in \overline{\mathbb{K}}^{m \times n}$ may be viewed as the set of its row vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$. If $\mathbf{B}$ is non-singular, then its set of row vectors form a basis.

5.2.2 Algebraic Background

Let $\mathbb{K}$ be a number field, i.e. an algebraic extension of $\mathbb{Q}$ of finite degree, and let d be its degree over $\mathbb{Q}$. Let $\cdot^\star$ denote the complex conjugation over $\mathbb{K}$, it is an involution and an automorphism of $\mathbb{K}$, which collapses to the identity if and only if $\mathbb{K}$ is a real number field. We let $\mathcal{R}$ be the ring of integers of $\mathbb{K}$ (its maximal order), $\mathbb{K}^+$ be the maximal real subfield of $\mathbb{K}$ and $\mathcal{R}^+ \subset \mathcal{R}$ be the ring of integers of $\mathbb{K}^+$ (we denote d^+ the degree of $\mathbb{K}^+$). We also define the completions $\overline{\mathbb{K}} = \mathbb{R} \otimes_{\mathbb{Q}} \mathbb{K}$ and $\overline{\mathbb{K^+}} = \mathbb{R} \otimes_{\mathbb{Q}} \mathbb{K}^+$. We note that those completions are not necessarily fields.

The number field $\mathbb{K}$ comes with d complex embeddings $\mathbb{K} \mapsto \mathbb{C}$ (forming a set $\mathfrak{S}$) , indexed by $i \in \{1, \ldots, d\}$. Similarly $\mathbb{K}^+$ comes with d^+ real embeddings $\mathbb{K}^+ \mapsto \mathbb{R}$ (forming a set $\mathfrak{S}^+$). Each σ in $\mathfrak{S}$ (resp. $\mathfrak{S}^+$) can be extended to the completion $\overline{\mathbb{K}}$ (resp. $\overline{\mathbb{K^+}}$). An element $e \in \overline{\mathbb{K}}$ (or $\overline{\mathbb{K^+}}$) is invertible if and only if all its embeddings are non-zero. Otherwise, e is said to be singular.

An element e of $\overline{\mathbb{K^+}}$ is said totally positive (and we write $e > 0$) if for all the real embeddings $\sigma \in \mathfrak{S}^+$ we have $\sigma(e) > 0$. Note that if e is totally positive, then it is invertible. If e is totally positive, it admits 2^{d^+} square roots in $\overline{\mathbb{K^+}}$, and we define its canonical square root $\sqrt{e} \in \overline{\mathbb{K^+}}$ as its unique square root that is totally positive : $\sqrt{e} > 0$. Note that this implies $\sigma(\sqrt{e}) = \sqrt{\sigma(e)}$ for all real embeddings $\sigma \in \mathfrak{S}^+$. This extends naturally to a definition of totally non-negative elements, noted $e \geqslant 0$. This also equips the field $\mathbb{K}^+$ (and its completion $\overline{\mathbb{K^+}}$) with a partial order: $e \geqslant e' \Leftrightarrow e - e' \geqslant 0$.

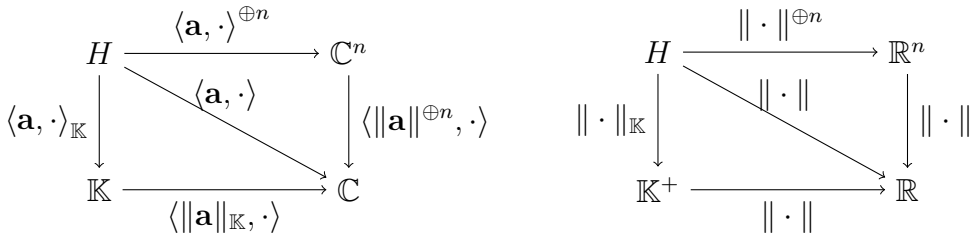
Hermitian structure of $\overline{\mathbb{K}}$. Seen as a $\mathbb{Q}$-vector space, $\mathbb{K}$ can be equipped with the sesquilinear map $\langle \cdot, \cdot \rangle : \mathbb{K} \times \mathbb{K} \to \mathbb{C}$, $(a, b) = \mathrm{Tr}(ab^\star) = \sum_\sigma \sigma(a)\sigma(b^\star)$. This sesquilinear map extends to $\overline{\mathbb{K}}$. The associated norm $x \mapsto \sqrt{\langle x, x \rangle} \in \mathbb{R}$ is noted $\| \cdot \|$.

Hermitian vector space over $\mathbb{K}$. For vector spaces $H = \mathbb{K}^n$, we can also define an sesquilinear product $\langle \cdot, \cdot \rangle_{\mathbb{K}} : H \times H \to \mathbb{K}$,

$$\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbb{K}} = \sum a_i b_i^\star.$$

One indeed verifies that $\langle \mathbf{a}, \mathbf{a} \rangle_{\mathbb{K}} \geqslant 0$ for any vector $\mathbf{a} \in \mathbb{K}$, and that $\langle \mathbf{a}, \mathbf{a} \rangle_{\mathbb{K}} \neq 0$ for any non-zero vector $\mathbf{a} \in \mathbb{K} \setminus \{\mathbf{0}\}$ (we carefully note that it does not imply that $\langle \mathbf{a}, \mathbf{a} \rangle_{\mathbb{K}} > 0$). The associated norm is given by $\| \cdot \|_{\mathbb{K}} : \mathbf{a} \mapsto \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle_{\mathbb{K}}}$. This map can be completed to $\overline{H} \times \overline{H} \to \overline{\mathbb{K}}$ where $\overline{H} = \mathbb{R} \otimes_{\mathbb{Q}} V$.

The two sesquilinear maps compose nicely: denoting $\langle \cdot, \cdot \rangle^{\oplus n}$ (resp. $\| \cdot \|^{\oplus n}$) the component-wise application of $\langle \cdot, \cdot \rangle$ (resp. $\| \cdot \|$), for all $a \in \mathbb{K}^n$ we have the following commutative diagrams



that naturally defines a sesquilinear map $H \times H \to \mathbb{C}$ together with a norm $\|\cdot\| : H \to \mathbb{R}$.

5.2.3 Lattice over a Ring

A lattice over the ring $\mathcal{R}$ is a discrete $\mathcal{R}$-module of $H = \mathbb{K}^n$ equipped with the Euclidean norm described above.

5.2.4 Gram-Schmidt Orthogonalization over Number Fields

Equipped with this algebraic background, we may now generalize the whole Section 2.1, in particular notions related to the Gram-Schmidt orthogonalization (GSO) of a basis. For a matrix $\mathbf{B} \in \mathbb{K}^{n \times m}$, the conjugate transpose of $\mathbf{B}$ is, as the name suggests, the $m \times n$ matrix $\mathbf{B}^\star$ whose coefficients verify $(\mathbf{B}^\star)_{ji} = (\mathbf{B}_{ji})^\star$.

Definition 5.1. *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_k\} \in H^n$ and $X \subseteq \mathbb{K}$. We note $\mathrm{Span}_X(\mathbf{b}_1, ..., \mathbf{b}_k)$ (or $\mathrm{Span}_X(\mathbf{B})$) the set $\{\sum_{1 \leqslant i \leqslant k} x_i \mathbf{b}_i, x_i \in X\}$. In particular, $\mathrm{Span}_{\mathcal{R}}(\mathbf{B})$ is an $\mathcal{R}$-module and and $\mathrm{Span}_{\mathbb{K}}(\mathbf{B})$ is a $\mathbb{K}$-vector space. If the vectors of $\mathbf{B}$ are linearly independent as elements of a $\mathbb{K}$-vector space, we say that $\mathbf{B}$ is a ($\mathbb{K}$-)basis (of $\mathrm{Span}_{\mathbb{K}}(\mathbf{B})$).*

We now have all the tools to generalize the Gram-Schmidt orthogonalization over vectors of $\mathbb{K}^m$ (instead of $\mathbb{R}^m$).

Definition 5.2 (Generalized GSO)**.** *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$. For any $k \in [\![1, n]\!]$, we note $V_k \triangleq \mathrm{Span}_{\mathbb{K}}(\mathbf{b}_1, ..., \mathbf{b}_k)$. The (generalized GSO) of $\mathbf{B}$ is the unique basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\} \in H^n$ verifying any of these equivalent properties:*

1. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \mathbf{Proj}(\mathbf{b}_k, V_{k-1})$
2. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k = \mathbf{b}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{b}_k, \tilde{\mathbf{b}}_j \rangle_{\mathbb{K}}}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathbb{K}}} \tilde{\mathbf{b}}_j$
3. $\forall k \in [\![1, n]\!], \tilde{\mathbf{b}}_k \perp V_{k-1}$ *and* $(\mathbf{b}_k - \tilde{\mathbf{b}}_k) \in V_{k-1}$

Noting $\tilde{V}_k \triangleq \mathrm{Span}_{\mathbb{K}}(\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_k)$, we also have: $\forall k \in [\![1, n]\!], \tilde{V}_k = V_k$.

As the matrix $\tilde{\mathbf{B}}$ in the previous lemma is uniquely defined, we can also generalize the Gram-Schmidt norm (Def. 2.19). Just as the usual Gram-Schmidt norm is useful for Klein's sampler, ours will be useful for its ring version.

Definition 5.3 (Generalized Gram-Schmidt norm)**.** *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in H^n$ be a basis, and $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$ its Gram-Schmidt orthogonalization. We call ($\mathbb{K}$-)Gram-Schmidt norm, and note $\left|\tilde{\mathbf{B}}\right|_{\mathbb{K}}$, the smallest value in $\mathbb{R}^+$ such that*

$$\forall i \in [\![1, n]\!], \left|\tilde{\mathbf{B}}\right|_{\mathbb{K}} \geqslant \|\tilde{\mathbf{b}}_i\|_{\mathbb{K}}$$

This definition subsumes and encompasses notions used in distinct Gaussian samplers. For $\overline{\mathbb{K}} = \mathbb{R}$, it matches the definition of the Gram-Schmidt norm from Definition 2.21 (and also from e.g. [ABB10b, AFV11, ADM12]). And for $n = m = 1$, the generalized Gram-Schmidt norm coincides with the definition of the largest singular value $s_1(\mathbf{B})$, used in [Pei10] to quantify the standard deviation of the output of Peikert's sampler. We recall that the largest singular value $s_1(\mathbf{B})$ of a real matrix $\mathbf{B}$ is defined by $s_1(\mathbf{B}) = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{xB}\|}{\|\mathbf{x}\|}$.

5.2.5 Gaussians

The norm $\|\cdot\| : \overline{\mathbb{K}}^n \to \mathbb{R}^+$ defined at the end of Subsection 5.2.2 also allows to generalize the entire Section 2.3. The Gaussian function $\rho : \overline{\mathbb{K}}^n \to (0, 1]$ is defined as follows:

$$\rho(\mathbf{x}) \triangleq \exp\left(-\|\mathbf{x}\|^2/2\right)$$

Replacing $\mathbb{R}$ with $\overline{\mathbb{K}}$, the rest of Subsection 2.3 is unchanged, except for the Lemma 2.32 which needs to be slightly rephrased:

Lemma 5.4 (Generalization of Lemma 2.32)**.** *Let $\mathbb{K}$ be a number field of degree d over $\mathbb{Q}$, and $\mathcal{R}$ be its ring of integers. For any $\epsilon \in (0, 1)$:*

$$\eta'_\epsilon(\mathcal{R}^n) \leqslant \frac{1}{\pi}\sqrt{\frac{1}{2}\log\left(2nd\left(1+\frac{1}{\epsilon}\right)\right)}$$

5.3 Ring Variants of Klein's and Peikert's Samplers

In this section, we present ring variants of Klein's and Peikert's samplers.

5.3.1 A Ring Variant of Klein's Sampler

Here we present a ring generalization of Klein's algorithm, where $\mathbb{Z}$ (resp. $\mathbb{Q}$ and $\mathbb{R}$) is replaced by $\mathcal{R}$ (resp. $\mathbb{K}$ and $\overline{\mathbb{K}}$). To get an intuition of why this could be faster than Klein's sampler, see that each output is of the form $\mathbf{v} = \sum_i z_i \mathbf{b}_i \in \mathcal{R}^m$, where the $z_i \in \mathcal{R}$. Then this algorithm samples an entire z_i at each step, whereas the original algorithm from Klein can only sample one coordinate of one z_i at each step.

Algorithm 5.1 RingKlein$(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}^{n\times m}$, its GSO $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\} \in \mathbb{K}^{n\times m}$, $\sigma \in \overline{\mathbb{K}}^+$, target $\mathbf{c} \in \overline{\mathbb{K}}^m$

Ensure: $\mathbf{v}$ sampled in a distribution close to $D_{\mathrm{Span}_\mathcal{R}(\mathbf{B}),\sigma,\mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$ $\quad \in \overline{\mathbb{K}}^m$
$\mathbf{v}_n \leftarrow \mathbf{0}$ $\quad \in \mathcal{R}^m$
for $i \leftarrow n, ..., 1$ **do**
  $d_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle_\mathbb{K} / \|\tilde{\mathbf{b}}_i\|^2_\mathbb{K}$ $\quad \in \overline{\mathbb{K}}$
  $\Sigma_i \leftarrow \sigma^2 / \|\tilde{\mathbf{b}}_i\|^2_\mathbb{K}$ $\quad \in \overline{\mathbb{K}}$
  $z_i \leftarrow \mathsf{Sample}(\mathcal{R}, \Sigma_i, d_i)$ $\quad \in \mathcal{R}$
  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \mathbf{b}_i$ $\quad \in \overline{\mathbb{K}}^m$
  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \mathbf{b}_i$ $\quad \in \mathcal{R}^m$
end for
return $\mathbf{v}_0$

In Algorithm 5.1, Sample is assumed to be a perfect discrete Gaussian sampler over $\mathcal{R}$: given a ring $\mathcal{R}$, a covariance $\Sigma \in \overline{\mathbb{K}^+}$ and a center $d \in \overline{\mathbb{K}}$, we assume $\mathsf{Sample}(\mathcal{R}, \Sigma, d) = D_{\mathcal{R},\sqrt{\Sigma},d}$. This mirrors the sampler in [GPV08], which uses a discrete Gaussian sampler over $\mathbb{Z}$ as an oracle. Here $\mathbb{Z}$ is replaced by $\mathcal{R}$, which of course raises practicality issues, but these questions will be addressed in Section 5.4.

The rest of this subsection is devoted to analyzing the correctness of Algorithm 5.1. Since the lemmas and their proofs are mostly identical to similar counterparts in [GPV08, DLP14], readers interested only in the practical applications may wish to acknowledge Theorem 5.7 and then skip to the next section.

Lemma 5.5. *For any input* $(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ *and output* $\mathbf{v} = \sum_i \hat{z}_i \mathbf{b}_i \in \mathrm{Span}_{\mathcal{R}}(\mathbf{B})$ *of* RingKlein*,*

$$\mathbf{v} - \mathbf{c} = \sum_i (\hat{z}_i - d_i)\tilde{\mathbf{b}}_i$$

where the values $c_i, \hat{z}_i$ *are as in* $\mathsf{RingKlein}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c}) \to \mathbf{v}$.

Proof. The proof is identical to the proof of Lemma 4.4 in [GPV08]. The only difference is that $\mathbb{Z}$ (resp. $\mathbb{R}$) is replaced with $\mathcal{R}$ (resp. $\overline{\mathbb{K}}$), and therefore $\Lambda(\mathbf{B})$ (resp. $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, ..., \mathbf{b}_k)$) has to be replaced with $\mathrm{Span}_{\mathcal{R}}(\mathbf{B})$ (resp. $\mathrm{Span}_{\overline{\mathbb{K}}}(\mathbf{b}_1, ..., \mathbf{b}_k)$). □

Lemma 5.6. *For any input* $(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ *and output* $\mathbf{v} = \sum_i \hat{z}_i \mathbf{b}_i \in \mathrm{Span}_{\mathcal{R}}(\mathbf{B})$ *of* RingKlein*, the probability that* $\mathbf{v}$ *is output is exactly*

$$\rho_{\sigma,\mathbf{c}}(\mathbf{v}) \cdot \prod_{1 \leqslant i \leqslant n} \frac{1}{\rho_{\sqrt{\Sigma_i},d_i}(\mathcal{R})}$$

Proof. For each i, the probability that $z_i = \hat{z}_i$ (conditioned on $z_j = \hat{z}_j$ for all $j > i$) is exactly $D_{\mathcal{R},\sqrt{\Sigma_i},d_i}(\hat{z}_i)$. Therefore the probability that $\mathbf{v}$ is output is

$$\prod_{1 \leqslant i \leqslant n} D_{\mathcal{R},\sqrt{\Sigma_i},d_i}(\hat{z}_i) = \frac{\prod_{1 \leqslant i \leqslant n} \rho_{\sqrt{\Sigma_i},d_i}(\hat{z}_i)}{\prod_{1 \leqslant i \leqslant n} \rho_{\sqrt{\Sigma_i},d_i}(\mathcal{R})}$$

In the expression above, the numerator is

$$\prod_{1 \leqslant i \leqslant n} \rho_{\sqrt{\Sigma_i},d_i}(\hat{z}_i) = \prod_{1 \leqslant i \leqslant n} \rho_\sigma\left((\hat{z}_i - d_i)\|\tilde{\mathbf{b}}_i\|_{\mathbb{K}}\right) = \rho_\sigma\left(\Sigma_i(\hat{z}_i - d_i)\tilde{\mathbf{b}}_i\right) = \rho_{\sigma,\mathbf{c}}(\mathbf{v})$$

The first equality comes from the fact that $\Sigma_i = \sigma^2/\|\tilde{\mathbf{b}}_i\|_{\mathbb{K}}^2$, the second one from the pairwise orthogonality of the $\tilde{\mathbf{b}}_i$'s, and the last one from Lemma 5.5. □

Theorem 5.7. *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}^{n \times m}$ *be a* $\mathcal{R}$*-basis,* $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\} \in \mathbb{K}^{n \times m}$ *its GSO,* $\mathbf{c} \in \mathcal{R}^m$*. Let* $\epsilon \in (0, \frac{1}{2n})$ *and* $\sigma \in \overline{\mathbb{K}^+}$ *such that* $\sigma \geqslant \eta'_\epsilon(\mathcal{R}) \cdot |\tilde{\mathbf{B}}|_{\mathbb{K}}$*. The statistical distance (resp. KL divergence) between* $D_{\mathrm{Span}_{\mathcal{R}}(\mathbf{B}),\sigma,\mathbf{c}}$ *and the output distribution of* $\mathsf{RingKlein}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ *is upper bounded by* $2n\epsilon$ *(resp.* $2\left(1 - \left(\frac{1+\epsilon}{1-\epsilon}\right)^n\right)^2 \approx 8n^2\epsilon^2$*).*

Proof. The proof is almost identical to the proof of Theorem 2 in [DLP14].

Let $\mathcal{P} = D_{\mathrm{Span}_{\mathcal{R}}(\mathbf{B}),\sigma,\mathbf{c}}$, and $\mathcal{Q}$ be the output distribution of $\mathsf{RingKlein}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$. By assumption, $\sqrt{\Sigma_i} \geqslant \eta'_\epsilon(\mathcal{R})$, so from Lemma 2.31 we can infer that $\rho_{\sqrt{\Sigma_i},d_i}(\mathcal{R}) \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \cdot \rho_{\sqrt{\Sigma_i}}(\mathcal{R})$. Applying Lemma 5.6 then gives us a relative error between $\mathcal{P}$ and $\mathcal{Q}$:

$$\forall \mathbf{v} \in \mathrm{Span}_{\mathcal{R}}(\mathbf{B}), |\mathcal{P}(\mathbf{v}) - \mathcal{Q}(\mathbf{v})| \leqslant \left(1 - \left(\frac{1+\epsilon}{1-\epsilon}\right)^n\right)\mathcal{P}(\mathbf{v})$$

At this point we get the statistical distance (resp. KL divergence) using a straightforward computation (resp. Lemma 3.5). □

5.3.2 A Ring Variant of Peikert's Sampler

In this subsection, we present and analyze a ring variant of Algorithm 1 from [Pei10]. Although we do not introduce new techniques to allow such a transformation (which was started in [Pei10] and completed in [DN12a]), we provide a complete description of a ring variant and give an analysis of its divergence from a perfect sampler.

For this step, we require a $\mathbb{Z}$-basis of the ring $\mathcal{R}$, that we denote by $(e_1, \dots, e_d)$. For example, if $\mathcal{R} = \mathbb{Z}[x]/(f(x))$ one may take a power-basis $1, x, \dots, x^{\deg(f)-1}$, but other choices may lead to better results (see [LPR13a]). One note that for the variance $\Sigma_e = \sum e_i e_i^\star \in \mathbb{K}^+$, it is essentially trivial to sample a distribution close to $D_{\mathcal{R},\eta\cdot\sqrt{\Sigma_e},c_i}$ (where $\eta = \eta_\epsilon(\mathbb{Z}) \in \mathbb{R}$), by computing $\sum_{1\leqslant i\leqslant d} a_i e_i$ where each a_i is drawn from $D_{\mathbb{Z},\eta,c_i}$ and $c = \sum_{1\leqslant i\leqslant d} c_i e_i$.

Algorithm 5.2 RingPeikert$(\mathcal{R}, \Sigma, c)$

Require: A variance $\Sigma \in \overline{\mathbb{K}^+}$, a target $c \in \overline{\mathbb{K}}$, a precomputed value $b \in \overline{\mathbb{K}}$ such that $\Sigma_e(bb^\star + \eta^2) = \Sigma$
Ensure: z sampled according to $D_{\mathcal{R},\sigma,c}$
1: $y \leftarrow b \cdot D_{\overline{\mathbb{K}},\sqrt{\Sigma_e}}$ $\quad \in \overline{\mathbb{K}}$
2: $z \leftarrow D_{\mathcal{R},\eta\sqrt{\Sigma_e},c+p}$ $\quad \in \mathcal{R}$
3: **return** z

This algorithm is a particular case of Peikert's sampler, unlike RingKlein which is a generalization of Klein's sampler. Therefore the analysis is much easier as we already studied Peikert's sampler in the general case (Theorem 2.34 and Lemma 3.7).

Theorem 5.8. *Let $\epsilon \in (0, \frac{1}{2})$, $b \in \overline{\mathbb{K}}$ such that $\Sigma_e(bb^\star + \eta^2) = \Sigma$ and $\eta \geqslant \eta'_\epsilon(\mathcal{R})$. The statistical distance (resp. KL divergence) between the output of* RingPeikert$(\mathcal{R}, \Sigma, \mathbf{c})$ *and $D_{\mathcal{R},\sqrt{\Sigma},\mathbf{c}}$ is upper bounded by $\approx 2\epsilon$ (resp. $\approx 8\epsilon^2$).*

Proof. Let us take $\Lambda_1 := \mathcal{R}$, $\Sigma_1 := \eta^2\Sigma_e$, $\Sigma_2 := b\bar{b}\Sigma_e$, $\mathbf{c} := c$, $\mathbf{y} := y$. We can apply Lemma 3.7 and conclude. □

5.4 Hybrid Algorithms for Sampling and Reduction

5.4.1 A Hybrid Sampler

In this section, we show that the two algorithms presented in Section 5.3 can be efficiently combined: more precisely, using RingPeikert as a subroutine of RingKlein will give us a new discrete Gaussian sampler.

Definition 5.9. *Using the notations from Section 5.3, we call Hybrid sampler and note* HybridSampler$(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ *the algorithm* RingKlein$(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ *where instead of calling a perfect sampler* Sample$(\mathcal{R}, \Sigma_i, d_i)$ *in step 6, we call* RingPeikert$(\mathcal{R}, \Sigma_i, d_i)$.

We now show that for carefully chosen parameters, the output distribution of the Hybrid sampler is close (in the sense of either the statistical distance or the KL divergence)

to a perfect discrete Gaussian. Given the nature of this sampler (it combines RingKlein and RingPeikert), we have to take into account the divergence of both RingKlein and RingPeikert from a "perfect behavior".

It is tempting to first quantify the divergence between HybridSampler and RingKlein, and then use the triangle inequality along with Theorem 5.7 to get the divergence between HybridSampler and a perfect discrete Gaussian. This approach works in the case of statistical distance but is useless for KL divergence since the latter does not verify the triangle inequality. Instead, we directly compute the statistical distance (resp. KL divergence) between both distributions.

Theorem 5.10. *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}^{m\times n}$ be a $\mathcal{R}$-basis, $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\} \in \mathbb{K}^{m\times n}$ its GSO and $\mathbf{c} \in \mathcal{R}^m$. Let $\epsilon \in (0, \frac{1}{6n})$ and $\sigma \in \overline{\mathbb{K}^+}$ such that $\sigma \geqslant \eta'_\epsilon(\mathcal{R}) \cdot |\tilde{\mathbf{B}}|_{\mathbb{K}}$.*

The statistical distance (resp. KL divergence) between $D_{\mathrm{Span}_\mathcal{R}(\mathbf{B}),\sigma,\mathbf{c}}$ and the output distribution of $\mathsf{HybridSampler}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ is upper bounded by $\frac{1}{2}((\frac{1+\epsilon}{1-\epsilon})^{3n} - 1) \approx 3n\epsilon$ (resp. $2((\frac{1+\epsilon}{1-\epsilon})^{3n} - 1)^2 \approx 72n^2\epsilon^2$).

Proof. This proof reprises elements from the proofs of Theorems 5.7 and 2.34.

Let $\mathcal{Q}$ be the output distribution of $\mathsf{HybridSampler}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$ and $\mathcal{P} = D_{\mathrm{Span}_\mathcal{R}(\mathbf{B}),\sigma,\mathbf{c}}$. Divergence between $\mathcal{P}$ and $\mathcal{Q}$ come from both the use of RingKlein and RingPeikert. Theorem 5.7 (resp. 5.8) quantifies the difference between the output of RingKlein (resp. RingPeikert) and a perfect Gaussian, upon the condition $\sigma \geqslant \eta'_\epsilon(\mathcal{R}) \cdot |\tilde{\mathbf{B}}|_{\mathbb{R}}$ (resp. $\sqrt{\Sigma_1} \geqslant \eta'_\epsilon(\Lambda_1)$). Coincidentally, in this case, the two conditions end up being exactly the same: the ϵ mentioned in Theorems 5.7 and 5.8 are actually the same here.

Let $\mathbf{v} = \sum_i \hat{z}_i \mathbf{b}_i \in \mathrm{Span}_\mathcal{R}(\mathbf{B})$. For any i, let $\mathcal{Q}_i$ be the output distribution of $\mathsf{RingPeikert}(\mathcal{R}, \Sigma_i, d_i)$, where the Σ_i, d_i are as in $\mathbf{v} \leftarrow \mathcal{Q}$. $\mathcal{Q}_i(\hat{z}_i) \in \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right] D_{\mathcal{R},\sqrt{\Sigma_i},d_i}$, as detailed in the proof of Theorem 2.34. Therefore

$$\begin{aligned}
\mathcal{Q}(\mathbf{v}) &= \textstyle\prod_{1\leqslant i\leqslant n} \mathcal{Q}_i(\hat{z}_i) \\
&\in \left[(\tfrac{1-\epsilon}{1+\epsilon})^n, (\tfrac{1+\epsilon}{1-\epsilon})^n\right] \textstyle\prod_i D_{\mathcal{R},\sqrt{\Sigma_i},d_i}(\hat{z}_i) \\
&\in \left[(\tfrac{1-\epsilon}{1+\epsilon})^n, (\tfrac{1+\epsilon}{1-\epsilon})^n\right] \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{v})}{\prod_i \rho_{\sqrt{\Sigma_i},d_i}(\mathcal{R})} \\
&\in \left[(\tfrac{1-\epsilon}{1+\epsilon})^n, (\tfrac{1+\epsilon}{1-\epsilon})^n\right] \left[1, (\tfrac{1+\epsilon}{1-\epsilon})^n\right] \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{v})}{\prod_i \rho_{\sqrt{\Sigma_i}}(\mathcal{R})}
\end{aligned}$$

Where the second equality comes from the fact that for each i, $\mathcal{Q}_i(\hat{z}_i) \in \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right] D_{\mathcal{R},\sqrt{\Sigma_i},d_i}$, the third one from Lemma 5.6 and the fourth from Lemma 2.31.

Let $\alpha = \frac{\rho_{\sigma,\mathbf{c}}(\mathrm{Span}_\mathcal{R}(\mathbf{B}))}{\prod_i \rho_{\sqrt{\Sigma_i}}(\mathcal{R})}$. We can then write $\mathcal{Q}(\mathbf{v}) \in \alpha \left[(\frac{1-\epsilon}{1+\epsilon})^n, (\frac{1+\epsilon}{1-\epsilon})^{2n}\right] \mathcal{P}(\mathbf{v})$. Summing $\mathcal{Q}(\mathbf{v})$ over $\mathrm{Span}_\mathcal{R}(\mathbf{B})$ yields

$$1 \in \alpha \left[\left(\frac{1-\epsilon}{1+\epsilon}\right)^n, \left(\frac{1+\epsilon}{1-\epsilon}\right)^{2n}\right] \overbrace{\sum_{\mathbf{v}\in\mathrm{Span}_\mathcal{R}(\mathbf{B})} \mathcal{P}(\mathbf{v})}^{=1}$$

This implies that $\alpha \in \left[(\frac{1-\epsilon}{1+\epsilon})^{2n}, (\frac{1+\epsilon}{1-\epsilon})^n\right]$, so we get a relative error bound between $\mathcal{P}$ and $\mathcal{Q}$:

$$|\mathcal{Q}(\mathbf{v}) - \mathcal{P}(\mathbf{v})| \leqslant \left(\left(\frac{1+\epsilon}{1-\epsilon}\right)^{3n} - 1\right) \mathcal{P}(\mathbf{v})$$

We can then conclude using a straightforward computation (resp. Lemma 3.5). □

The bound on the statistical distance (resp. KL divergence) can then be used to assert the security of the scheme following a standard argument (resp. Lemma 3.3).

5.4.2 Hybrid Babai and Size-Reduction Algorithms

One could see Babai's rounding (resp. nearest plane) algorithm as a specific instantiation of Klein's (resp. Peikert's) sampler for a standard deviation $\sigma = 0$. While Babai's algorithms [Bab85, Bab86] were invented long before the aforementioned samplers and do not serve the same purpose, it is nevertheless correct to view them like this, under the convention that a Gaussian with standard deviation 0 behaves like an exact rounding. Therefore, the method used to create a hybrid sampler from Klein's and Peikert's sampler can be used to create a hybrid approximation algorithm for the closest vector problem from Babai's approximation algorithms, which can also be seen as a ring generalization of algorithm NearestPlane.

Definition 5.11. *Let $\mathbf{B} \in \mathcal{R}^{m \times n}$ be a $\mathcal{R}$-basis, $\tilde{\mathbf{B}} =$ its GSO and $\mathbf{c} \in \mathcal{R}^m$. We define*

$$\mathsf{HybridBabai}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \cdot) \triangleq \mathsf{HybridSampler}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, 0, \cdot)$$

We now give a bound on the output of the algorithm. The proof can be done either by generalizing the proof of Babai's nearest plane algorithm (Proposition 2.26) or by reprising the proof of Theorem 5.10.

Lemma 5.12. *For any $\mathbf{c} \in \overline{\mathbb{K}}^m$, $\mathsf{HybridBabai}(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \mathbf{c})$ outputs a point $\mathbf{v}$ verifying:*

$$\mathbf{v} - \mathbf{c} = \sum_{1 \leqslant i \leqslant n} \varepsilon_i \tilde{\mathbf{b}}_i$$

where $\varepsilon_i \in \overline{\mathbb{K}}$ are such that, for all $i \leqslant n$, $-\frac{1}{2} \leqslant \varepsilon_i \leqslant \frac{1}{2}$ (as inequalities of $\overline{\mathbb{K}}$).

Since size-reducing a basis of n elements consists of $n-1$ sequential executions of NearestPlane on this basis, this means we can also generalize the size-reduction over $\mathcal{R}$-modules. We then get a size-reduction algorithm for basis over $\mathcal{R}$. A very similar and actually slightly more generic algorithm already exists in [FS10, Figure 3], so we do not elaborate further on this topic.

5.5 Precision Analysis of the Hybrid Sampler

In this section, we run a precision analysis of our sampler. Indeed, using it in exact arithmetic is very costly, if ever possible (the subroutine samples from a continuous Gaussian), so in practice it has to be executed floating-point arithmetic.

To simplify our analysis, we analyze our sampler for $\mathcal{R}$-lattices where $\mathcal{R} = \mathbb{Z}[x]/(x^k+1)$: all the operation over $\mathcal{R}$-vectors and $\mathcal{R}$-matrices can then be described using "usual" matrices and vectors over $\mathbb{R}$, and are less prone to error propagation than rings such as $\mathbb{Z}[x]/(\phi_p(x)) = \mathbb{Z}[x]/(x^{p-1} + ... + x + 1)$ for a prime p (for these rings, one solution is to embed them in $\mathbb{Z}[x]/(x^p - 1)$). We also consider that the center $\mathbf{c}$ has integer coefficients, as it does in practice. To clarify the setting to readers, here is how the Hybrid sampler can be described when $\mathcal{R} = \mathbb{Z}[x]/(x^k + 1)$:

Algorithm 5.3 Hybrid$(\mathcal{R}, \mathbf{B}, \tilde{\mathbf{B}}, \Sigma, \mathbf{c})$

Require: Basis $\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_n\} \in \mathbb{Z}^{kn \times km}$, its block GSO $\tilde{\mathbf{B}} = \{\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n\} \in \mathbb{Q}^{kn \times km}$, $\Sigma \in \mathbb{Q}^{k \times k}$, target $\mathbf{c} \in \mathbb{Z}_q^{km}$

Ensure: $\mathbf{v}$ sampled in a distribution close to $D_{\mathrm{Span}_{\mathcal{R}}(\mathbf{B}), \sqrt{\Sigma}, \mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$ $\quad \in \mathbb{Z}^{km}$
$\mathbf{v}_n \leftarrow \mathbf{0}$ $\quad \in \mathbb{Z}^{km}$
for $i \leftarrow n, ..., 1$ **do**
  $\mathbf{d}_i \leftarrow \mathbf{c}_i \tilde{\mathbf{B}}_i^+$ $\quad \in \mathbb{Q}^k$
  $\Sigma_i \leftarrow \Sigma(\tilde{\mathbf{B}}_i \tilde{\mathbf{B}}_i^t)^{-1}$ $\quad \in \mathbb{Q}^{k \times k}$
  $\mathbf{x}_i \leftarrow D_1^k$ $\quad \in \mathbb{R}^k$ } PeikertSampler subroutine:
  $\mathbf{y}_i \leftarrow \mathbf{x}_i \cdot \mathbf{C}_i$, where $\mathbf{C}_i = \sqrt{\Sigma_i - r^2 \mathbf{I}_k}$ $\quad \in \mathbb{R}^k$ } $\mathbf{z}_i \leftarrow D_{\mathbb{Z}^k, \sqrt{\Sigma_i}, \mathbf{d}_i}$
  $\mathbf{z}_i \leftarrow \lfloor \mathbf{d}_i - \mathbf{y}_i \rceil_r$ $\quad \in \mathbb{Z}^k$ }
  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - \mathbf{z}_i \mathbf{B}_i$ $\quad \in \mathbb{Z}^{km}$
  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + \mathbf{z}_i \mathbf{B}_i$ $\quad \in \mathbb{Z}^{km}$
end for
return $\mathbf{v}_0$

Each of the matrices $\mathbf{B}_i, \tilde{\mathbf{B}}_i$ is a row concatenation of m anticirculant matrices. By the canonical isomorphism between $\mathcal{R}$ and $\mathcal{A}_N(\mathcal{R})$ (resp. between $\mathbb{K}$ and $\mathcal{A}_N(\mathbb{K})$), the matrices $\mathbf{B}_i, \tilde{\mathbf{B}}_i$ may be viewed as the matrix form of the vectors $\mathbf{b}_i, \tilde{\mathbf{b}}_i$ of Algorithm 5.1.

As the hybrid sampler mixes Klein's and Peikert's samplers, its precision analysis also borrows elements from the analyses of both algorithms. Readers interested into understanding the details of the proof of Lemma 5.13 should be familiar with the proof of Lemma 3.16. It thoroughly explains some computations whose details are omitted in the proof of Lemma 5.13 to keep the proof from being unnecessarily long.

Lemma 5.13. *Let $k, m, n, q \in \mathbb{N}^\star$ such that $k \geqslant 128$. Let $\epsilon \in (0, 1/4)$, $\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_n\} \in \mathbb{Z}^{kn \times km}$ a basis, $\tilde{\mathbf{B}} = \{\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n\}$ its block GSO, $\Sigma \in \mathbb{Q}^{k \times k}$ and a target $\mathbf{c} \in \mathbb{Z}_q^{km}$. For $i \in [\![1, n]\!]$, we note $\mathbf{D}_i \triangleq \tilde{\mathbf{B}}_i^+$. Let $r \geqslant \eta'_\epsilon(\mathbb{Z}^k)$ such that $\forall i \in [\![1, n]\!], \Sigma > r^2(\tilde{\mathbf{B}}_i \tilde{\mathbf{B}}_i^t)$. Let $\delta \in [0, .01)$, $(\bar{\mathbf{C}}_i, \bar{\mathbf{D}}_i)_i$ such that for each $i \in [\![1, n]\!]$:*

- $s_1(\mathbf{C}_i - \bar{\mathbf{C}}_i) \leqslant \delta \cdot s_1(\mathbf{C}_i)$
- $s_1(\mathbf{D}_i - \bar{\mathbf{D}}_i) \leqslant \delta \cdot s_1(\mathbf{D}_i)$

Let $\mathcal{E}$ denote Algorithm 5.3 using perfectly precomputed values $(\mathbf{C}_i, \mathbf{D}_i)_i$ and a perfect oracle $\mathcal{N}_0$ to sample $\mathbf{x}_i \leftarrow D_1^k$. Similarly, $\bar{\mathcal{E}}$ denotes Algorithm 5.3 using (possibly imperfect) precomputed values $(\bar{\mathbf{C}}_i, \bar{\mathbf{D}}_i)_i$ and an oracle $\mathcal{N}_\delta$ (see Definition 3.14) to sample $\mathbf{x}_i \leftarrow D_1^k$. In addition, let $\mathcal{D}$ (resp. $\bar{\mathcal{D}}$) be the output distribution of $\mathcal{E}$ (resp. $\bar{\mathcal{E}}$). Finally, let:

$$\delta_c = \delta \sqrt{k} \cdot \max_i [q \sqrt{m} s_1(\mathbf{D}_i) + 3 s_1(\mathbf{C}_i)]$$

$$C = \frac{n}{r^2} \cdot \delta_c \cdot \left[r\sqrt{2\pi k} + \frac{r\sqrt{2\pi k}\epsilon}{1 - \epsilon} + \delta_c \right]$$

The KL divergence between $\mathcal{D}$ and $\bar{\mathcal{D}}$ is bounded as follows:

$$D_{KL}(\bar{\mathcal{D}} \| \mathcal{D}) \leqslant (e^C - 1)^2$$

Remark 5.14. Once again, the expression of the bound for the KL divergence can be simplified for practical parameters: if $\delta, \epsilon \ll 1$, then

$$e^C - 1 \approx C \approx \frac{2.5nk\delta}{r} \cdot \max_i [q\sqrt{m}s_1(\mathbf{D}_i) + 3s_1(\mathbf{C}_i)].$$

Proof. The goal of this proof is to bound the ratio $\mathcal{D}(\mathbf{v})/\bar{\mathcal{D}}(\mathbf{v})$ for an overwhelmingly large (in terms of probability) set of possibly outputs $\mathbf{v} \in \mathcal{L}(\mathbf{B})$.

Let $\hat{\mathbf{v}} = \sum_i \hat{\mathbf{z}}_i \mathbf{B}_i$ be a potential output of Algorithm 5.3. During the execution of $\mathcal{E}$, each $\hat{\mathbf{z}}_i$ is sampled from $\lfloor \mathbf{d}_i - \mathbf{y}_i \rceil_r$, where $\mathbf{d}_i = \mathbf{c}_i \mathbf{D}_i$, $\mathbf{y}_i = \mathbf{x}_i \mathbf{C}_i$ and $\mathbf{x}_i$ is sampled from D_1^k. Similarly, during the execution of $\bar{\mathcal{E}}$, each $\hat{\mathbf{z}}_i$ is sampled from $\lfloor \bar{\mathbf{d}}_i - \bar{\mathbf{y}}_i \rceil_r$, where $\bar{\mathbf{d}}_i = \mathbf{c}_i \bar{\mathbf{D}}_i$, $\bar{\mathbf{y}}_i = \bar{\mathbf{x}}_i \bar{\mathbf{C}}_i$ and $\bar{\mathbf{x}}_i$ is sampled from D_1^k then rounded to a relative precision at least δ.

For any i, let $\mathcal{D}_i$ (resp. $\bar{\mathcal{D}}_i$) denote the output distribution of $\lfloor \mathbf{d}_i - \mathbf{y}_i \rceil_r$ (resp. $\lfloor \bar{\mathbf{d}}_i - \bar{\mathbf{y}}_i \rceil_r$), so that we have $\mathcal{D}(\hat{\mathbf{v}}) = \prod_i \mathcal{D}_i(\hat{\mathbf{z}}_i)$ (resp. $\bar{\mathcal{D}}(\hat{\mathbf{v}}) = \prod_i \bar{\mathcal{D}}_i(\hat{\mathbf{z}}_i)$). For the reason explained in Remark 3.15, we have $\forall i, \|\mathbf{x}_i\|, \|\bar{\mathbf{x}}_i\| \leqslant \sqrt{2k}$ with overwhelming probability. Since $\bar{\mathbf{x}}_i$ approximates $\mathbf{x}_i$ with relative precision at least δ, we also have $\|\bar{\mathbf{x}}_i - \mathbf{x}_i\| \leqslant \delta \|\mathbf{x}_i\| \leqslant \delta\sqrt{2k}$. Therefore:

$$\begin{aligned} \mathbf{y}_i - \bar{\mathbf{y}}_i &= \mathbf{x}_i \mathbf{C}_i - \bar{\mathbf{x}}_i \bar{\mathbf{C}}_i \\ &= \mathbf{x}_i(\mathbf{C}_i - \bar{\mathbf{C}}_i) + (\mathbf{x}_i - \bar{\mathbf{x}}_i)\bar{\mathbf{C}}_i \\ \Rightarrow \quad \|\mathbf{y}_i - \bar{\mathbf{y}}_i\| &\leqslant 2.1\sqrt{2k}\delta s_1(\mathbf{C}_i) \end{aligned} \tag{5.1}$$

which is straightforward from the triangle inequality, the fact that $\|\mathbf{x}\mathbf{C}\| \leqslant \|\mathbf{x}\| s_1(\mathbf{C})$ and that $s_1(\bar{\mathbf{C}}_i) \leqslant s_1(\mathbf{C}_i) + s_1(\bar{\mathbf{C}}_i - \mathbf{C}_i) \leqslant (1+\delta)s_1(\mathbf{C}_i) \leqslant 1.1 s_1(\mathbf{C}_i)$. As an immediate consequence:

$$\begin{aligned} \|(\mathbf{d}_i - \mathbf{y}_i) - (\bar{\mathbf{d}}_i - \bar{\mathbf{y}}_i)\| &\leqslant \|\mathbf{d}_i - \bar{\mathbf{d}}_i\| + \|\mathbf{y}_i - \bar{\mathbf{y}}_i\| \\ &\leqslant \delta[q\sqrt{km}s_1(\mathbf{D}_i) + 2.1\sqrt{2k}s_1(\mathbf{C}_i)] \\ &\leqslant \delta_c \end{aligned} \tag{5.2}$$

Using the exact same reasoning and techniques as in the proof of Lemma 3.16, we have:

$$\forall i, A \leqslant \log\left(\frac{\mathcal{D}_i(\hat{\mathbf{z}})}{\bar{\mathcal{D}}_i(\hat{\mathbf{z}})}\right) \leqslant B \tag{5.3}$$

Where $|A| \leqslant \frac{1}{r^2} \cdot \delta_c \cdot \left[r\sqrt{2\pi k} + \frac{r\sqrt{2\pi k}\epsilon}{1-\epsilon}\right]$ and $|B| \leqslant \frac{1}{r^2} \cdot \delta_c \cdot \left[r\sqrt{2\pi k} + \frac{r\sqrt{2\pi k}\epsilon}{1-\epsilon} + \delta_c\right]$. We have $|A|, |B| \leqslant C/n$, therefore combining the relations $\mathcal{D}(\hat{\mathbf{v}}) = \prod_i \mathcal{D}_i(\hat{\mathbf{z}}_i)$, $\bar{\mathcal{D}}(\hat{\mathbf{v}}) = \prod_i \bar{\mathcal{D}}_i(\hat{\mathbf{z}}_i)$ and equation 5.3 yields:

$$e^{-C} \leqslant \frac{\mathcal{D}(\hat{\mathbf{v}})}{\bar{\mathcal{D}}(\hat{\mathbf{v}})} \leqslant e^C \tag{5.4}$$

Which allows to conclude using Lemma 3.5. □

5.6 Trade-offs Using Subfields

Let us assume that we are working in a number field $\mathbb{K}$ of degree n that admits a subfield $\mathbb{J}$ of degree $b|n$, and let $d = n/b$ be the relative degree of $\mathbb{K}$ over $\mathbb{J}$. The field $\mathbb{K}$ may then be seen as a sub-algebra of the algebra of matrices $\mathbb{J}^{d\times d}$. That way, a basis $\mathbf{B} \in \mathbb{K}$ may be seen as a matrix of $\mathbb{J}^{dk\times dk}$, and one may run our algorithm over $\mathbb{J}$ rather than $\mathbb{K}$ to improve the quality of our sampler, at the price of slowing it down. At the extreme case, we may choose $\mathbb{J} = \mathbb{Q}$ in which case we are simply back to running the original algorithm of Klein.

Such a trade-off exists for all cyclotomic number fields of non-prime conductor. Indeed, if $\mathbb{K}_n$ denotes the n-th cyclotomic number field, then $\mathbb{K}_d$ is a subfield of $\mathbb{K}_n$ if and only if d divide n.

Because such an algebraic description is not exactly straightforward to translate into an implementation, in the next section we show explicitly how to realize this trade-off when n is a power of 2 and $d = n/2$.

5.6.1 Explicit Trade-off for power-of-two Cyclotomic Fields

In this subsection, m will be a power of two, $N = \varphi(m) = m/2$, $\mathbb{K}_m = \mathbb{Q}[x]/(\phi_m) = \mathbb{Q}[x]/(x^N + 1)$ and $\mathcal{Z}_m = \mathbb{Z}[x]/(\phi_m)$. The reader will notice, that, in this case, the decomposition as a matrix over a subfield is extremely similar to the steps of a Fast Fourier Transform.

Let $f \in \mathbb{K}_m$ and suppose we want to sample a spherical Gaussian over the lattice $\mathrm{Span}_{\mathcal{Z}_m}(f)$. This lattice is either generated by the N-dimensional $\mathbb{Z}$-basis $\mathbf{B} \triangleq \mathcal{A}_N(f)$ or by the 1-dimensional $\mathcal{Z}_m$-basis f.

Using the Hybrid sampler here seems pointless at first sight, since we end up with Peikert's sampler if we use $\mathcal{Z}_m$ as base ring (i.e. $\mathbb{K}_m$ as base field).

Fortunately $\mathbb{K}_m$ is a field extension of $\mathbb{K}_{m/2}$: $\mathbb{K}_m \cong (\mathbb{K}_{m/2})^2$. More practically, if one notes $f(x) = f_1(x^2) + xf_2(x^2)$, a simple permutation of rows and columns transforms $\mathbf{B}$ into another matrix $\mathbf{B}'$ with a structure over $\mathbb{K}_{m/2}$:

$$P_\pi^t \times \mathbf{B} \times P_\pi = \left[\begin{array}{c|c} \mathcal{A}_{\phi_m}(f_1) & \mathcal{A}_{\phi_m}(f_2) \\ \hline \mathcal{A}_{\phi_m}(xf_2) & \mathcal{A}_{\phi_m}(f_1) \end{array}\right] \triangleq \mathbf{B}'$$

Where P_π is the permutation matrix associated to π, defined as:

$$\begin{cases} \pi(2k+1) = k+1 \\ \pi(2k) = \frac{N}{2} + k + 1 \end{cases}$$

Now we can use the Hybrid Sampler with the basis $\mathbf{B}'$ using $\mathcal{Z}_{m/2}$ as a base ring instead of $\mathcal{Z}_m$. This process can of course be recursively iterated: one then obtains a basis with a ring structure over $\mathcal{Z}_{m/4}$, then $\mathcal{Z}_{m/8}$, and so on, down to $\mathbb{Z}$, which corresponds to Klein's sampler.

This is of course a trade-off: while breaking down $\mathcal{Z}_m$ in smaller rings allows to sample with a smaller standard deviation, the running time of the new sampler can be up to twice longer, as $\mathcal{Z}_{m/2}$ is twice as small as $\mathcal{Z}_{m/2}$ but the basis now contains four times more ring elements. In Chapter 7, taking a different representation will allow to avoid this caveat and push this idea to its limit.

Figure 5.2 illustrates this trade-off: f is split between its even and odd coefficients, and commuting the basis elements separates them.

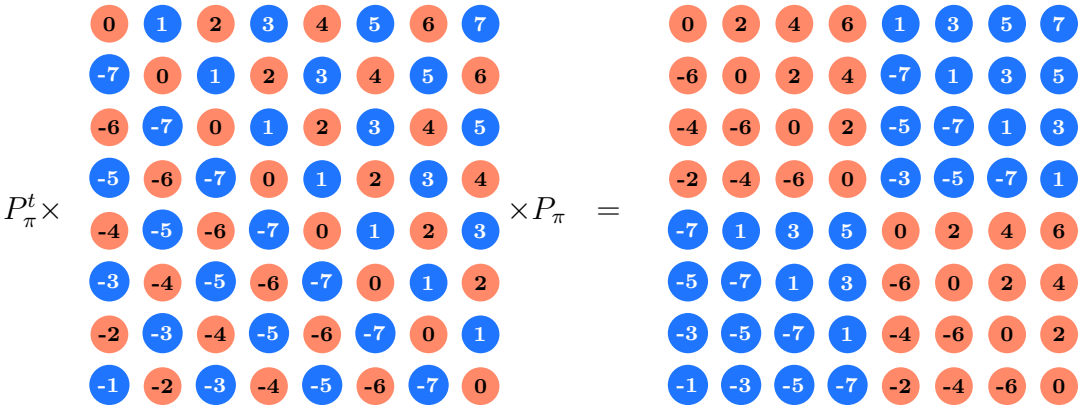


Figure 5.2: An example of trade-off: using P_π as a change-of-basis matrix, with $\pi = \left(\begin{smallmatrix}1&2&3&4&5&6&7&8\\1&5&2&6&3&7&4&8\end{smallmatrix}\right)$, allows us to turn a $\mathbb{Z}[x]/(x^8+1)$-basis into a $\mathbb{Z}[x]/(x^4+1)$-basis with twice as many elements.

Chapter 6

Full-Domain-Hash Signatures over NTRU Lattices

6.1 Introduction

In the previous chapter, we devised a Hybrid algorithm for performing Gaussian sampling over $\mathcal{R}$-lattices for some ring $\mathcal{R}$. This algorithm mixes Klein's and Peikert's samplers, and as a result it yields a speed-quality trade-off between them. The goal of the current chapter is twofold.

We perform a detailed instantiation of the Full-Domain Hash (FDH) signature scheme from [GPV08], which we instantiate over the class of NTRU lattices. While proper use of lattice trapdoors [GPV08] is quite recent, the design of compact lattices with quite a good trapdoor comes from the NTRU-cryptosystems [HPS98, HHGP+03]. Unlike more recent constructions [MP12], those lattices are not necessarily uniformly random (unless one makes impractical choices of parameters [SS11]), in which case they may not benefit from theoretical worst-case hardness [Ajt96, MR07]. Nevertheless, when it comes to practical instantiation and concrete security, this type of construction remains to date the most efficient one. The signature scheme that we obtain yields very compact signatures, whose sizes are competitive with those of the BLISS [DDLL13] signature scheme. However, the practical efficiency of this scheme still remains to be evaluated. While we expect the verification running time will be similar to BLISS', the signing procedure will very likely be much slower.

Our instantiation of the FDH signature scheme of [GPV08] uses Gaussian sampling as a central procedure. This is where we can compare the three samplers. Each of them yields a different speed-security trade-off. Extensive experiments backed up by heuristics strongly suggest that for fixed parameters $N = 512, q = 1024$, Klein's sampler attains a security of 192 bits, against 160 for our Hybrid sampler and 120 bits for Peikert's. So security-wise, we are closer to Klein's sampler. But on the efficiency front, our sampler is $\tilde{O}(n)$ times faster than Klein's sampler and only about twice slower than Peikert's sampler. While not being optimal on both fronts, our Hybrid sampler is closer in quality to Klein's sampler, and much closer to Peikert's sampler.

6.1.1 Roadmap

This chapter is organized as follows. We first give complements on NTRU lattices in Section 6.2. Then Section 6.3 details the signature scheme, gives parameters and summarizes our results. For readers interested in the specificities of each sampler, Section 6.4 detail briefly how the results were obtained for each of them. Additionally, Section 6.5 gives heuristic explanations and asymptotic estimates to the quality gap between the different samplers.

6.2 Complements on NTRU Lattices

NTRU lattices were briefly defined in Section 4.7 but we give a more formal definition of them in this section. In the rest of the chapter, $m \in \mathbb{N}^\star$ will be a power of two, $q \in \mathbb{N}^\star$, $N = \varphi(m)$, $\mathcal{Z} = \mathbb{Z}[x]/(\phi_m(x)) = \mathbb{Z}[x]/(x^N + 1)$ and $\mathcal{K} = \mathbb{Q}[x]/(\phi_m(x))$.

Definition 6.1 (NTRU Lattices)**.** *Let $f, g, F, G \in \mathcal{Z}$ be such that*

$$fG - gF = q \bmod (x^N + 1) \tag{6.1}$$

Then the NTRU lattice generated by f, g, F, G is the lattice generated by the rows of the block matrix

$$\mathbf{B}_{f,g,F,G} = \left[\begin{array}{c|c} \mathcal{A}(g) & -\mathcal{A}(f) \\ \hline \mathcal{A}(G) & -\mathcal{A}(F) \end{array}\right]$$

Where $\mathcal{A}(p) = \mathcal{A}_{\phi_m}(p)$ is the $N \times N$ matrix which i-th row is the coefficients of $x^{i-1} \cdot p(x) \bmod (x^N + 1)$.

How to find (F, G) given (f, g) is not important for us since it is all done in the key generation. It can be achieved efficiently [HHGP+03] and we describe one way (which is not really new) of doing it in the ulterior Section 8.3, Algorithm 8.1. Moreover, for any fixed $(g, -f)$ and distinct $(G_1, -F_1)$, $(G_2, -F_2)$ verifying equation 6.1, $\Lambda(\mathbf{B}_{f,g,F_1,G_1}) = \Lambda(\mathbf{B}_{f,g,F_2,G_2})$ so in the rest of the chapter we can simply assume that $(G, -F)$ is reduced with respect to $\mathrm{Span}_{\mathcal{R}}((g, -f))$ using RoundOff (Algorithm 2.2) and note $\mathbf{B}_{f,g} \triangleq \mathbf{B}_{f,g,F,G}$.

Some properties of anticirculant matrices $\mathcal{A}(\cdot)$ are given in Section 4.2.4. In addition, when m is a power of two, the embedding norm $\mathbb{Q}[x]/(\phi_m(x))$ coincides with the usual norm and $\mathcal{A}(p)^t = \mathcal{A}(f^\star)$, which makes many operations simpler and more transparent.

Proposition 6.2. *Let $f, g, F, G \in \mathcal{R}$ verifying equation 6.1 and $\mathbf{h} = gf^{-1} \bmod q$. Then $\mathbf{A}_{\mathbf{h},q} = \begin{pmatrix} -\mathcal{A}(\mathbf{h}) & \mathbf{I}_N \\ q\mathbf{I}_N & \mathbf{O}_N \end{pmatrix}$ and $\mathbf{B}_{f,g} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$ generate the same lattice*

$$\Lambda_{\mathbf{h},q} = \{(\mathbf{u}, \mathbf{v}) \in \mathcal{Z}^2 | \mathbf{u} + \mathbf{v}\mathbf{h} = 0 \bmod q\}$$

Proof. Consider $\mathbf{P} = \mathbf{A}_{\mathbf{h},q} \times \mathbf{B}_{f,g}^{-1}$ the change-of-basis matrix between $\mathbf{A}_{\mathbf{h},q}$ and $\mathbf{B}_{f,g}$. One can check that $q\mathbf{P} = O_{2N} \bmod q$, so $\mathbf{P} \in \mathbb{Z}^{2N \times 2N}$. Also, $|\det(\mathbf{P})| = 1$ so $\mathbf{P}^{-1} \in \mathbb{Z}^{2N \times 2N}$. We can conclude that $\mathbf{A}_{\mathbf{h},q}$ and $\mathbf{B}_{f,g}$ both generate the same lattice. □

If N, q are high enough and f, g are generated with enough entropy, we expect $\mathbf{h}$ to be computationally indistinguishable from random and f, g to be hard to recover. $\mathbf{B}_{f,g}$ and $\mathbf{A}_{\mathbf{h},q}$ then become ideal building blocks (as a private and public key) for building public key cryptographic primitives.

A Note on NTRU Signatures

The history of NTRU lattices is very interesting. As stated by Lyubashevsky [Lyu12a], their construction naturally follows from a consequence of a series of recent works on (ring) lattice-based cryptography [Reg05, LPR10, SS11]. Yet the original construction [HPS98] predates all these works.

If the original NTRUEncrypt [HPS98] encryption scheme has successfully resisted years of cryptanalysis and remains to date the most efficient public-key encryption key over lattices, it has taken longer to obtain a secure signature scheme. Various attempts to do Hash-and-Sign schemes on NTRU lattices [HPS01, HHGP+03, eHGH+08] have been subsequently broken [GJSS01, GS02, NR06, DN12b]. Only recently have provably secure constructions [GPV08, Lyu12b] led to secure signature schemes over NTRU lattices [SS11, ABDG14], including the most efficient known signature scheme over lattices [DDLL13].

Sampling over NTRU Lattices

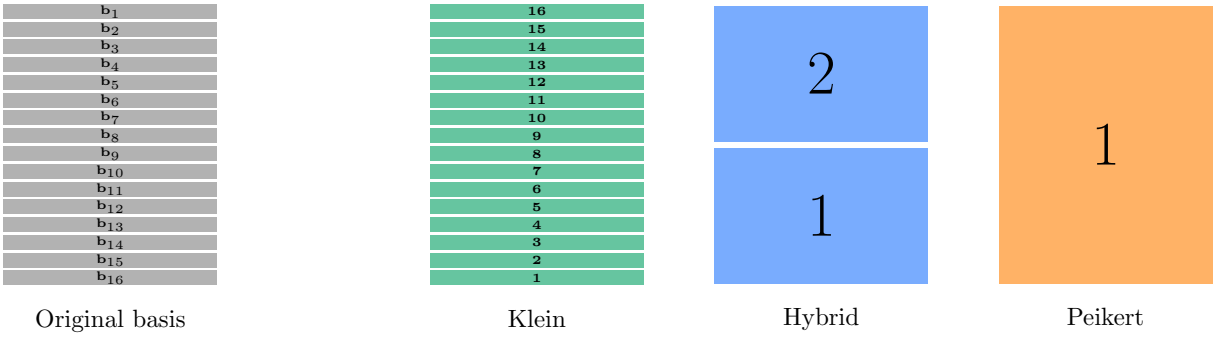
NTRU lattices are not only $\mathbb{Z}$-modules of dimension $2N$, but also $\mathcal{Z}$-modules of dimension 2. This non-trivial structure is what allows us to use our Hybrid sampler. Depending if we take $\mathbb{Z}$, $\mathcal{Z}$ or $\mathcal{Z}^{2\times 2}$ as our base ring $\mathcal{R}$, we can do Gaussian sampling in three different ways:

1. Taking $\mathcal{R} = \mathbb{Z}$, our hybrid algorithm is simply the original Klein's sampler. This approach is developed in Subsection 6.4.1.

2. Taking $\mathcal{R} = \mathcal{Z}$, we get a sampler that is strictly different from Klein's and Peikert's sampler. Indeed, $\mathcal{L}(\mathbf{B}_{f,g})$ can be seen as a $\mathcal{Z}$-module of rank 2. This approach is developed in Subsection 6.4.2.

3. Taking $\mathcal{R} = \mathcal{Z}^{2\times 2}$, one gets Peikert's sampler.[1] This approach is developed in Subsection 6.4.3.

The Figure 6.1 explicits the differences between each approach.



Each sampler outputs a vector $\mathbf{v} = \sum_i z_i \mathbf{b}_i$. The z_i's are computed one by one for Klein's sampler (the figure gives the order in which they are computed), all at the same time for Peikert's, and N at a time for our instantiation of the Hybrid sampler over NTRU lattices.

Figure 6.1: High Level Differences between the Gaussian Samplers

[1] To formally subsume that case with our algorithm, we would need to generalize our description to non-commutative rings.

6.3 Methodology and Global Results

To compare the three samplers, we instantiate over NTRU lattices the provably secure FDH signature scheme proposed in [GPV08], which is also described in Section 2.5.1, Figure 2.6.

Algorithm 6.1 $\mathsf{KeyGen}(N, q)$

Require: N, q
Ensure: Private key $\mathbf{B} \in \mathbb{Z}_q^{2N \times 2N}$ and public key $\mathbf{h} \in \mathcal{Z}_q$
1: $f, g \leftarrow \mathcal{D}$ for some distribution $\mathcal{D}$ over $\mathcal{Z}$
2: Compute $F, G \in \mathcal{Z}$ such that $fG - gF = q \bmod (x^N + 1)$
3: $\mathbf{h} = gf^{-1} \bmod q$
4: $\mathbf{B} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$
5: $\mathsf{SK} \leftarrow \mathbf{B}, \mathsf{PK} \leftarrow \mathbf{h}$

Here, the signing key $\mathbf{B}$ is a short basis orthogonal to $\mathbf{A} = \left[\mathbf{I} | \mathcal{A}(\mathbf{h})^t\right]$, making it a trapdoor for sampling short elements $(\mathbf{s}_1, \mathbf{s}_2)$ such that $\mathbf{s}_1 + \mathbf{s}_2\mathbf{h} = \mathbf{t}$ for any $\mathbf{t}$, without leaking any information about itself. We voluntarily do not precise which distribution $\mathcal{D}$ we use in step 1, since this will depend on which sampler we choose.

Algorithm 6.2 $\mathsf{Sign}(\mathbf{B}, \mathbf{m})$

Require: Private key $\mathbf{B} \in \mathbb{Z}_q^{2N \times 2N}$, hash function $H : \{0,1\}^\star \to \mathcal{Z}_q$, standard deviation σ, message $\mathbf{m}$
Ensure: Signature $\mathbf{s}(\mathbf{m}) \in \mathcal{Z}_q$
1: $\mathbf{t} \leftarrow H(\mathbf{m}) \in \mathcal{Z}_q$
2: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow (\mathbf{t}, \mathbf{0}) - \mathsf{Sample}(\mathbf{B}, \sigma, (\mathbf{t}, \mathbf{0}))$ \\ $\mathbf{s}_1 + \mathbf{s}_2\mathbf{h} = \mathbf{t}$
3: $\mathbf{s}(\mathbf{m}) \leftarrow (\mathbf{s}_1, \mathbf{s}_2)$
4: **return** $\mathbf{s}(\mathbf{m})$

In the Sign algorithm, we have omitted the statefulness for clarity reasons. Moreover, the step 2 uses as a black box an algorithm Sample such that the output of $\mathsf{Sample}(\mathbf{B}, \sigma, \mathbf{c})$ is statistically close $D_{\mathcal{L}(\mathbf{B}), \sigma, \mathbf{c}}$. This is where we plug in either Klein's, the Hybrid or Peikert's sampler.

Algorithm 6.3 $\mathsf{Verify}(\mathbf{h}, \mathbf{m})$

Require: Public key $\mathbf{h} \in \mathcal{Z}_q$, hash function $H : \{0,1\}^\star \to \mathcal{Z}_q$, message $\mathbf{m}$
Ensure: Signature $(\mathbf{s}_1, \mathbf{s}_2) \in \mathcal{Z}_q^2$
1: $\mathbf{t} \leftarrow H(\mathbf{m}) \in \mathcal{Z}_q$
2: **if** $\mathbf{s}_1 + \mathbf{s}_2\mathbf{h} = \mathbf{t}$ **and** $\|(\mathbf{s}_1, \mathbf{s}_2)\| \geqslant 2\sigma\sqrt{\pi N}$ **then**
3: **accept**
4: **else**
5: **reject**
6: **end if**

We now fix the parameters $N = 512, q = 2^{20}$, and consider the FDH signature scheme describes in the algorithms 6.1, 6.2 and 6.3. We try to find optimal NTRU bases – security-wise – for each sampler. As the security decreases when the standard deviation increases,

we know *a priori* that Klein's will yield the highest security level and Peikert's the lowest, due to equation 6.2.

$$\sqrt{q} = \det(\mathbf{B}_{f,g})^{\frac{1}{2N}} \leqslant \left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathbb{R}} \leqslant \left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}} \leqslant s_1(\mathbf{B}_{f,g}) \tag{6.2}$$

What we want is to quantify these security levels, which is equivalent to compute $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathbb{R}}$ for Klein's sampler, $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ for the Hybrid and $s_1(\mathbf{B}_{f,g})$ for Peikert's. This is what we do in Section 6.4.

Once it is done, we use the techniques in [GN08, CN11, DDLL13] to evaluate the concrete security of the schemes. The way lattice schemes are analyzed is to determine the hardness of the underlying lattice problem, which is measured using the "root Hermite factor" introduced in [GN08]. If one is looking for a vector $\mathbf{v}$ in an n-dimensional lattice that is larger than the n^{th} root of the determinant, then the associated root Hermite factor is

$$\frac{\|\mathbf{v}\|}{\det(\Lambda)^{1/n}} = \gamma^n \tag{6.3}$$

If one is looking for an unusually-short planted vector $\mathbf{v}$ in an NTRU lattice, then the associated root Hermite factor, according to the experiments in [DDLL13] is

$$\frac{\sqrt{n/(2\pi e)} \cdot \det(\Lambda)^{1/n}}{\|\mathbf{v}\|} = .4\gamma^n \tag{6.4}$$

Based on the results in [GN08, CN11], one can get a very rough estimate of the hardness of the lattice problem based on the value of γ (unfortunately, there has not been enough lattice cryptanalysis literature to have anything more than just a rough estimate). For values of $\gamma \approx 1.007$, finding the vector is at least 2^{80}-hard. For values less that 1.004, the problem seems completely intractable and is approximated to be at least 192-bits hard.

The results of our experiments are summarized in the Table 6.1.

Table 6.1: Comparison of the three approaches for NTRU lattices of fixed dimension $2N = 1024$ and modulus $q = 2^{20}$

Sampler	**Klein**	**Hybrid**	**Peikert**
Security level	**192**	**160**	**120**
Root Hermite factor γ	1.0042	1.0049	1.0060
Ring $\mathcal{R}$	$\mathbb{Z}$	$\mathcal{Z}$	$\mathcal{Z}^{2\times 2}$
$\text{rank}_{\mathcal{R}}(\Lambda)$	$2N$	2	1
Form of $\mathbf{B}$	$\mathcal{R}^{2N\times 2N}$	$\mathcal{R}^{2\times 2}$	$\mathcal{R}^{1\times 1}$
Running time	$\tilde{O}(N^2)$	$\tilde{O}(N)$	$\tilde{O}(N)$
Smallest $\left\|\tilde{\mathbf{B}}\right\|_{\mathcal{R}}$ attained	$1.17\sqrt{q}$	$2.5\sqrt{q}$	$8\sqrt{q}$

One can see that in terms of standard deviation (and therefore of security), the hybrid sampler is closer to Klein's sampler than to Peikert's sampler. However, its runtime (assuming fast floating-point arithmetic operations) is much closer to the one of Peikert's sampler. In this setting the hybrid sampler is a trade-off between Klein's and Peikert's samplers, however it manages to be close to getting the best out of each algorithm.

Moreover, Section 6.5 indicates that both phenomenons amplify when the dimension increases: the security gets even closer to Klein's than Peikert's, and the other way around for the running time.

6.4 Sampling over NTRU Lattices: Bounds and Experiments

6.4.1 Klein's Sampler ($\mathcal{R} = \mathbb{Z}, \mathbb{K} = \mathbb{Q}$)

We know that the norm of vectors sampled by Klein's Sampler is proportional to $|\tilde{\mathbf{B}}_{f,g}|$. This section analyses how to compute efficiently $|\tilde{\mathbf{B}}_{f,g}|$. This is done by Corollary 6.5, which is a direct consequence of Lemmas 6.3 and 6.4.

Lemma 6.3. *Let $\mathbf{B}_{f,g}$ be a NTRU basis, and $\mathbf{b}_1, ..., \mathbf{b}_{2N}$ be the row vectors of $\mathbf{B}_{f,g}$. Then $|\tilde{\mathbf{B}}_{f,g}| = max\{\|\tilde{\mathbf{b}}_1\|, \|\mathbf{b}_{N+1}\|\}$*

Proof. $\mathbf{B}_{f,g}$ is a block isometric basis (of block size N) with respect to the isometry $r : (u, v) \in \mathcal{K}^2 \mapsto (x \cdot u, x \cdot v)$. Therefore applying Algorithm 4.4 to it yields its GSO. As a subroutine, Algorithm 4.4 uses Algorithm 4.2 on $\{\mathbf{b}_1, ..., \mathbf{b}_N\}$ and $\{\tilde{\mathbf{b}}_{N+1}, ..., r^{N-1}(\tilde{\mathbf{b}}_{N+1})\}$. As a result, $\|\tilde{\mathbf{b}}_1\| \geqslant ... \geqslant \|\tilde{\mathbf{b}}_N\|$ and $\|\tilde{\mathbf{b}}_{N+1}\| \geqslant ... \geqslant \|\tilde{\mathbf{b}}_{2N}\|$ (implicit from the recursive formula in Lemma 4.9). The result follows. □

Instead of computing $2N$ values $\|\tilde{\mathbf{b}}_1\|, ..., \|\tilde{\mathbf{b}}_{2N}\|$, there is now only two of them to compute. The following lemma gives an exact expression for $\|\tilde{\mathbf{b}}_{N+1}\|$.

Lemma 6.4. $\|\tilde{\mathbf{b}}_{N+1}\| = \left\|\left(\frac{qf^\star}{ff^\star+gg^\star}, \frac{qg^\star}{ff^\star+gg^\star}\right)\right\|$

Proof. $\tilde{\mathbf{b}}_{N+1}$ is given by the formula of Proposition 2.17:

$$\tilde{\mathbf{b}}_{N+1} = \mathbf{b}_{N+1} - \mathbf{b}_{N+1}\mathbf{B}_0^+\mathbf{B}_0$$

Where $\mathbf{B}_0 = [\mathcal{A}(g)| - \mathcal{A}(f)]$. The result then follows from the fact that $\mathcal{A}(f)^t = \mathcal{A}(f^\star)$, circulant matrices commute and $fG - gF = q \bmod (x^N + 1)$. □

Combining Lemmas 6.3 and 6.4 yield a formula for $|\tilde{\mathbf{B}}_{f,g}|$.

Corollary 6.5. *Let $\mathbf{B}_{f,g}$ be a NTRU basis. The classical Gram-Schmidt norm of $\mathbf{B}_{f,g}$ is:*

$$\left|\tilde{\mathbf{B}}_{f,g}\right| = \max\left(\|(g, -f)\|, \left\|\left(\frac{qf^\star}{ff^\star + gg^\star}, \frac{qg^\star}{ff^\star + gg^\star}\right)\right\|\right)$$

We ran experiments and computed a heuristic (see Section 6.5.1) to evaluate how small $|\tilde{\mathbf{B}}_{f,g}|$ can be made. Both indicate that the optimal choice for $\|\mathbf{b}_1\|$ is $\|\mathbf{b}_1\| \approx \sqrt{\frac{qe}{2}}$, since we then get $\|\tilde{\mathbf{b}}_{N+1}\| \approx \|\tilde{\mathbf{b}}_1\|$. Interestingly, neither show any correlation between the dimension N and the minimal $|\tilde{\mathbf{B}}_{f,g}|$. Moreover, $|\tilde{\mathbf{B}}_{f,g}|$ is very close to the theoretical lower bound $\sqrt{q}$ (see equation 6.2).

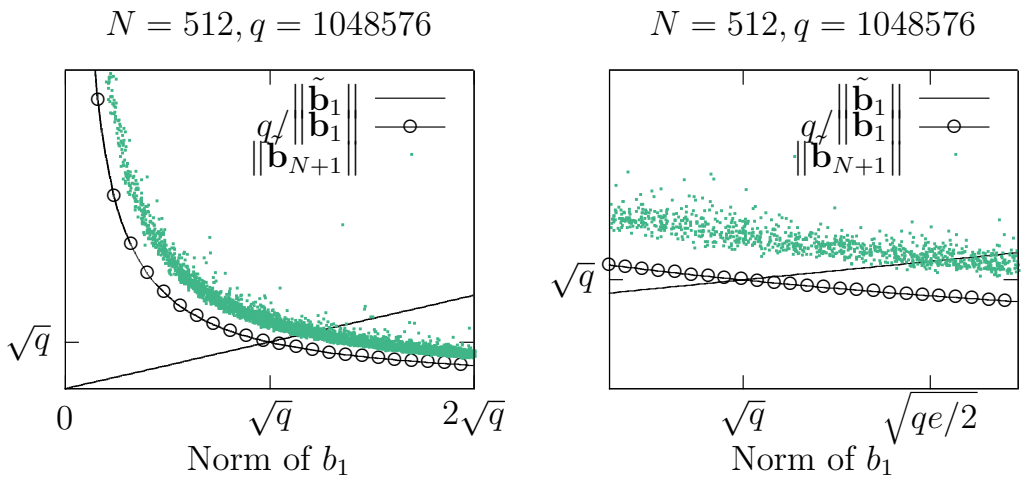


Figure 6.2: Values of candidates $\|\tilde{\mathbf{b}}_{N+1}\|$ and $\|\tilde{\mathbf{b}}_1\|$ for $|\tilde{\mathbf{B}}_{f,g}|$, with $N = 512, q = 2^{20}$. $q/\|\mathbf{b}_1\|$ is the lower bound for $\|\tilde{\mathbf{b}}_{N+1}\|$ given in Lemma 6.6.

The following lemma provides a theoretical lower bound for $\|\tilde{\mathbf{b}}_{N+1}\|$, given q and $\|\mathbf{b}_1\|$. In our case, $\|\tilde{\mathbf{b}}_{N+1}\|$ is very close to its lower bound.

Lemma 6.6. *Let $\mathbf{B} = (\mathbf{b}_i)_{1 \leqslant i \leqslant 2N}$ be a NTRU basis. $\|\tilde{\mathbf{b}}_{N+1}\|$ admits the following lower bound: $\|\tilde{\mathbf{b}}_{N+1}\| \geqslant q/\|\mathbf{b}_1\|$. As a corollary, $|\tilde{\mathbf{B}}| \geqslant \sqrt{q}$.*

Proof. We have $|\det(\mathbf{B}_{f,g})| = |\det(\tilde{\mathbf{B}}_{f,g})| = \prod_{i=1}^{N} \|\tilde{\mathbf{b}}_i\|$. We know that $|\det(\mathbf{B}_{f,g})| = q^N$ and that for any k in $[\![1; N]\!]$, $\|\tilde{\mathbf{b}}_i\| \leqslant \|\tilde{\mathbf{b}}_1\|$ and $\|\tilde{\mathbf{b}}_{N+i}\| \leqslant \|\tilde{\mathbf{b}}_{N+1}\|$. So $q^N \leqslant \|\mathbf{b}_1\|^N \|\tilde{\mathbf{b}}_{N+1}\|^N$. □

6.4.2 Hybrid Sampler ($\mathcal{R} = \mathcal{Z}, \mathbb{K} = \mathcal{K}$)

In the case where $\mathcal{R} = \mathcal{Z}$, the vectors $\mathbf{f} = (g, -f), \mathbf{F} = (G, -F) \in \mathcal{R}^2$ then generate $\Lambda(\mathbf{B}_{f,g})$ as a $\mathcal{R}$-module of rank 2: $\Lambda(\mathbf{B}_{f,g}) = \text{Span}_{\mathcal{R}}(\mathbf{f}, \mathbf{F})$.[2] We then get

Lemma 6.7. *Let $\mathbf{B}_{f,g}$ be a NTRU basis. The $\mathcal{K}$-Gram-Schmidt norm of $\mathbf{B}_{f,g}$ is:*

$$\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}} = \max_{\omega \in \Omega_m} \max\left(D(\omega), \frac{q}{D(\omega)}\right)$$

Where $D(x) \triangleq \|(f,g)\|_{\mathcal{K}}(x) (= \sqrt{|f(x)|^2 + |g(x)|^2}$ when $x \in \Omega_m$).

Proof. $\mathbf{B} = \{\mathbf{f} = (g, -f), \mathbf{F} = (G, -F)\}$ is the $\mathcal{R}$-basis used for this sampler. $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ is by definition the smallest value in $\mathbb{R}^+$ verifying $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}} \geqslant \mathbf{f}$ and $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}} \geqslant \tilde{\mathbf{F}}$. A straightforward computation using Proposition 2.17 gives us

$$\tilde{\mathbf{F}} = \left(\frac{qf^\star}{ff^\star + gg^\star}, \frac{qg^\star}{ff^\star + gg^\star}\right)$$

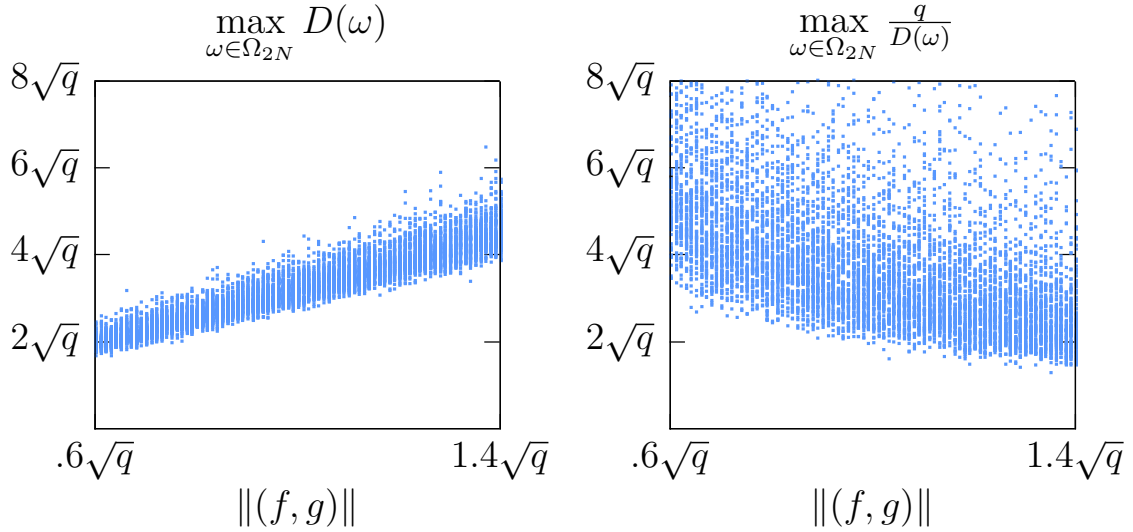
Therefore

$$\langle \tilde{\mathbf{F}}, \tilde{\mathbf{F}} \rangle_{\mathcal{K}} = \frac{q^2}{ff^\star + gg^\star} = \frac{q^2}{\langle \mathbf{f}, \mathbf{f} \rangle_{\mathcal{K}}}$$

[2] Note that in this setting, the equation 6.1 becomes $\det[\mathbf{f}, \mathbf{F}] = q \in \mathcal{R}$.

Evaluating $\langle \mathbf{f}, \mathbf{f} \rangle_{\mathcal{K}}$ and $\langle \tilde{\mathbf{F}}, \tilde{\mathbf{F}} \rangle_{\mathcal{K}}$ on all the $\omega \in \Omega_m$ then yields the result. □

We then generate random NTRU bases to give an estimate on the maximum Gram-Schmidt norm given by Lemma 6.7. When $\|\mathbf{f}\|$ increases, so do the values $(D(\omega))_{\omega \in \Omega_m}$, but the values $(\frac{q}{D(\omega)})_{\omega \in \Omega_m}$ decrease. The goal is to sample $\mathbf{f}$ with a standard deviation chosen so that $\max_{\omega \in \Omega_m}(D(\omega)) \approx \max_{\omega \in \Omega_m}(\frac{q}{D(\omega)})$, which should yield a reasonable small value for $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$. Figure 6.3 summarizes these experiments for NTRU bases of size $2N = 1024$.



$\mathbf{B}_{f,g}$ is a NTRU basis of dimension $2N = 1024$, for a modulus $q = 2^{20}$. On the left, $\max_{\omega \in \Omega_{2N}} D(\omega)$ and on the right, $\max_{\omega \in \Omega_{2N}} \frac{q}{D(\omega)}$, where $D(\omega)$ is defined as in Lemma 6.7.

Figure 6.3: The potential values of $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$.

In our experiments, we managed to get $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}} \approx 2.5\sqrt{q}$. This is just 2.5 times the theoretical smallest value that $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ can take. Also, a heuristic in Section 6.5, backed up by experiments, suggest that $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ is minimized for the same choice of parameters than $|\tilde{\mathbf{B}}_{f,g}|$, and that we then have $|\tilde{\mathbf{B}}_{f,g}|_{\mathcal{K}} \approx \alpha\sqrt{\log N}|\tilde{\mathbf{B}}_{f,g}|$ for some constant α that we evaluate to be $\alpha \approx 1.30$. This means that $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ can be made as small as approximately $1.52\sqrt{\log N} \cdot \sqrt{q}$.

6.4.3 Peikert's Sampler (Informally $\mathcal{R} = \mathcal{Z}^{2\times 2}, \mathbb{K} = \mathcal{K}^{2\times 2}$)

We recall that $s_1(\mathbf{B})$ is the largest singular value of $\mathbf{B}$. The standard deviation of the discrete Gaussian output by Peikert's sampler is $\eta'_\epsilon(\mathbb{Z}^{2N}) \cdot s_1(\mathbf{B}_{f,g})$. The following lemma gives the value of $s_1(\mathbf{B}_{f,g})$ for a NTRU basis.

Lemma 6.8. *Let $\mathbf{B}_{f,g}$ be a NTRU basis. The maximal singular value of $\mathbf{B}_{f,g}$ is*

$$s_1(\mathbf{B}_{f,g}) = \sqrt{\max_{\omega \in \Omega_m}(\lambda_\omega)}$$

where $C(x) \triangleq (ff^\star + gg^\star + FF^\star + GG^\star)(x)$ and $\lambda_\omega \triangleq \frac{1}{2}\left(C(\omega) + (-1)^i\sqrt{C^2(\omega) - 4q^2}\right)$.

Proof. The matrices $\mathcal{A}(f), \mathcal{A}(g), \mathcal{A}(F), \mathcal{A}(G)$ are co-diagonalizable in an orthonormal eigenbasis: there exists a matrix $\mathbf{P} \in \mathbb{C}^{N \times N}$ and a diagonal matrix $\mathbf{\Phi}_m \triangleq \underset{\phi_m(\zeta)=0}{\text{diag}}\ (\zeta)$ such that $\mathcal{A}(f) = \mathbf{P} \times f(\mathbf{\Phi}_m) \times \mathbf{P}^\star$. We can therefore write

$$\mathbf{B}_{f,g} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \times \begin{bmatrix} g(\mathbf{\Phi}_m) & -f(\mathbf{\Phi}_m) \\ G(\mathbf{\Phi}_m) & -F(\mathbf{\Phi}_m) \end{bmatrix} \times \begin{bmatrix} \mathbf{P}^\star & \mathbf{0} \\ \mathbf{0} & \mathbf{P}^\star \end{bmatrix}$$

The singular values of $\mathbf{B}_{f,g}$ are the positive square roots of the eigenvalues of $\mathbf{B}_{f,g}\mathbf{B}_{f,g}^t$. Since similar matrices share the same eigenvalues, we are therefore looking for the square roots of the eigenvalues of

$$\mathbf{M} = \begin{bmatrix} (ff^\star + gg^\star)(\mathbf{\Phi}_m) & (fF^\star + gG^\star)(\mathbf{\Phi}_m) \\ (f^\star F + g^\star G)(\mathbf{\Phi}_m) & (FF^\star + GG^\star)(\mathbf{\Phi}_m) \end{bmatrix}$$

We compute the characteristic polynomial of $\mathbf{M}$:

$$\begin{aligned} \chi_{\mathbf{M}}(\lambda) &= \det[\mathbf{M} - \lambda I_{2N}] \\ &= \det\left[q^2 I_N - \lambda(ff^\star + gg^\star + FF^\star + GG^\star)(\mathbf{\Phi}_m) + \lambda^2 I_N\right] \\ &= \prod_{\omega \in \Omega_m} (q^2 - \lambda(ff^\star + gg^\star + FF^\star + GG^\star)(\omega) + \lambda^2) \end{aligned}$$

The second equality first uses the fact that $\det \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \det[\mathbf{AD} - \mathbf{BC}]$ when $\mathbf{B}$ and $\mathbf{C}$ commute, then the equation 6.1, at at last the fact that for any $\omega \in \Omega_m$, $(ff^\star)(\omega) = |f(\omega)|^2$.

Noting $C(\omega) \triangleq (ff^\star + gg^\star + FF^\star + GG^\star)(\omega)$, the eigenvalues are, for $(\omega, i) \in (\Omega_m, \{0, 1\})$, the values

$$\lambda_{\omega,i} = \frac{1}{2}\left(C(\omega) + (-1)^i \sqrt{C^2(\omega) - 4q^2}\right)$$

Noticing that the maximal $\lambda_{\omega,i}$ must have $i = 0$, this concludes the proof. □

Just like in Subsection 6.4.2, we ran experiments on random NTRU bases to evaluate how small we could get the singular value $s_1(\mathbf{B}_{f,g})$ to be. The results are summarized in Figure 6.4.

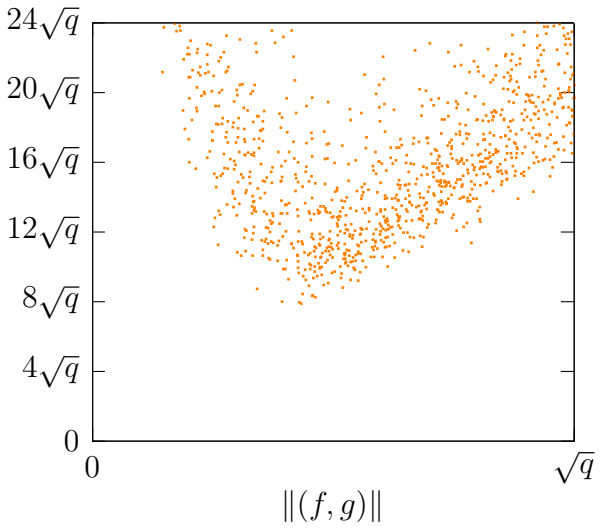


Figure 6.4: Values of $s_1(\mathbf{B}_{f,g})$, where $\mathbf{B}_{f,g}$ is a NTRU basis of dimension $2N = 1024$, for a modulus $q = 2^{20}$.

The best values we managed to get in our experiments were $s_1(\mathbf{B}_{f,g}) \approx 8\sqrt{q}$. Although this could seem to be "not too big" compared to $\left|\tilde{\mathbf{B}}_{f,g}\right|$ and $\left|\tilde{\mathbf{B}}_{f,g}\right|_{\mathcal{K}}$ (which condition the quality of the vectors output by Klein's and our Hybrid sampler), in practice it decreases the security parameter of the Full-Domain Hash signature scheme using Peikert's sampler. Once again, Section 6.5 provides a heuristic consistent with experiments that indicate that $s_1(\mathbf{B}_{f,g})$ can in practice be made as small as $1.52\left(\frac{N}{3}\right)^{1/4}\sqrt{\log N}\cdot\sqrt{q} \approx 1.15N^{1/4}\sqrt{\log N}\cdot\sqrt{q}$.

6.5 Sampling over NTRU Lattices: Heuristics and Asymptotics

This section is the conclusion of the previous one. In section 6.2, we compared the hybrid/Klein's/Peikert's samplers over NTRU lattices and found out that our hybrid sampler can sample with a standard deviation much closer to Klein's sampler than Peikert's. However, one could arguably call this a stroke of luck and question the validity of this assumption in high dimensions. In this section, we give heuristics to estimate how small the standard deviation can be set for each sampler *in high dimensions*. These heuristics are consistent with our experiments.

For convenience, we omit the smoothing parameter factor $\eta'_\epsilon(\mathbb{Z}^{2N})$ in the standard deviation σ. The heuristics indicate that we can sample with σ around:

- $1.17\sqrt{q}$ for Klein's sampler
- $1.52\sqrt{\log N}\cdot\sqrt{q}$ for our hybrid sampler
- $1.15N^{1/4}\sqrt{\log N}\cdot\sqrt{q}$ for Peikert's sampler

Two trends emerge from these heuristics:

- There exist a large subset of NTRU lattices for which Klein's sampler can be used with σ very close to its best known theoretical bound. More precisely, Klein's sampler can securely use a σ proportional to the value $|\tilde{\mathbf{B}}_{f,g}|$ which is known to be at least $\sqrt{q}$ for NTRU lattices, and we can get $|\tilde{\mathbf{B}}_{f,g}|$ to be at most $1.17\sqrt{q}$. Therefore, if one is not concerned with the speed of sampling, then Klein's sampler is the best choice.
- Regarding σ, there is a factor $O(\sqrt{\log N})$ (with a small constant) between our sampler and Klein's, compared to a factor $O(N^{1/4})$ between ours and Peikert's. So the fact that we got a security (and more precisely a root Hermite factor γ) closer to the one offered by Klein's sampler wasn't a fluke but a phenomenon that amplifies as N increases.

The heuristics can be found in Section 6.5.1 for Klein's sampler, and Section 6.5.2 for the hybrid and Peikert's samplers.

6.5.1 Optimal Bases for Klein's Sampler

We now a heuristic argument explaining why $|\tilde{\mathbf{B}}_{f,g}| = \max(\|\tilde{\mathbf{b}}_1\|, \|\tilde{\mathbf{b}}_{N+1}\|)$ is minimized for $\|\mathbf{b}_1\| \approx \sqrt{\frac{qe}{2}}$. We first give a lemma that will be useful for our heuristic.

Lemma 6.9. *Let $(\ell_1, \ell_2) \in \mathbb{R}_+^2$, and the two following sets of $(\mathbb{R}^N)^2 \cong \mathbb{R}^{2N}$:*

- $E_1 = \{(\mathbf{x}_1, \mathbf{x}_2) \in (\mathbb{R}^N)^2 | \|\mathbf{x}_1\| = \ell_1, \|\mathbf{x}_2\| = 0\}$.
- $E_2 = \{(\mathbf{x}_1, \mathbf{x}_2) \in (\mathbb{R}^N)^2 | \|\mathbf{x}_1\| = 0, \|\mathbf{x}_2\| = \ell_2\}$.

Let $\mathbf{v}_1 ... \mathbf{v}_N \xleftarrow{\$} E_1$, $\mathbf{w}_1 ... \mathbf{w}_N \xleftarrow{\$} E_2$. *Then:*

$$\mathbb{E}[\det(\mathbf{v}_1, ...\mathbf{v}_N, \mathbf{w}_1, ...\mathbf{w}_N)] = \frac{(2N)!}{(2N^N)N!} \|\mathbf{v}_1\|^N \|\mathbf{w}_N\|^N$$

Proof. We first recall the definition of the Gram matrix:

$$Gram(\mathbf{v}_1 ... \mathbf{v}_k) \triangleq (\langle \mathbf{v}_i, \mathbf{v}_j \rangle)_{1 \leqslant i,j \leqslant k},$$

and the Gram determinant:

$$G(\mathbf{v}_1 ... \mathbf{v}_k) \triangleq \det\left(Gram(\mathbf{v}_1 ... \mathbf{v}_k)\right)$$

For a full-rank basis, the Gram determinant verifies $G(\mathbf{v}_1 ... \mathbf{v}_k) = \det(\mathbf{v}_1 ... \mathbf{v}_k)^2$. We can therefore write:

$$\det(\mathbf{v}_1, ...\mathbf{v}_N, \mathbf{w}_1, ...\mathbf{w}_N)^2 = G(\mathbf{v}_1, ...\mathbf{v}_N, \mathbf{w}_1, ...\mathbf{w}_N) = G(\mathbf{v}_1, ...\mathbf{v}_N) \times G(\mathbf{w}_1, ...\mathbf{w}_N)$$

the second equality coming from the fact that the vectors $\mathbf{v}_i$ are orthogonal to the vectors $\mathbf{w}_j$. $G(\mathbf{v}_1, ...\mathbf{v}_N) = G(\tilde{\mathbf{v}}_1, ...\tilde{\mathbf{v}}_N) = \prod_{i=1}^N \|\tilde{\mathbf{v}}_i\|^2$, and since each $\tilde{\mathbf{v}}_i$ is the projection of the vector $\mathbf{v}_i$ over a subspace of dimension $2N - i + 1$, we have $\mathbb{E}[\|\tilde{\mathbf{v}}_i\|] = \sqrt{\frac{2N-i+1}{2N}} \|\mathbf{v}_i\|$. The same reasoning applies to the vectors $\mathbf{w}_j$, and we can conclude. □

Heuristic. We now use lemma 6.9 to estimate the value of $\|\mathbf{b}_1\|$ for which $\max(\|\tilde{\mathbf{b}}_1\|, \|\tilde{\mathbf{b}}_{N+1}\|)$ is minimized. For $i \in [\![1, N]\!]$, we denote:

$$\hat{\mathbf{b}}_{N+i} = \mathbf{b}_{N+i} - \mathbf{Proj}(\mathbf{b}_{N+i}, \mathrm{Span}(\mathbf{b}_1 ... \mathbf{b}_N)).$$

Since the basis $\mathbf{B}_{f,g}$ is block isometric, $\forall i \in [\![1; N]\!]$, $\|\hat{\mathbf{b}}_{N+i}\| = \|\tilde{\mathbf{b}}_{N+1}\|$ and:

$$\det(\mathbf{b}_1 ... \mathbf{b}_{2N}) = \det(\mathbf{b}_1, ...\mathbf{b}_N, \hat{\mathbf{b}}_{N+1}, ..., \hat{\mathbf{b}}_{2N}) = q^N.$$

Since the vectors $\mathbf{b}_i$ are orthogonal to the $\hat{\mathbf{b}}_{N+i}$, there exists an orthonormal basis of $\mathbb{R}^{2N}$ in which the N first coordinates of each $\mathbf{b}_i$ and the N last coordinates of each $\hat{\mathbf{b}}_{N+i}$ are zero. Plus, the $\mathbf{b}_i$ (resp. the $\hat{\mathbf{b}}_{N+i}$) have same norm so we assume they are distributed with respect to the distribution E_1 (resp. E_2) of lemma 6.9 with $\ell_1 = \|\tilde{\mathbf{b}}_1\|$ (resp. $\ell_2 = \|\tilde{\mathbf{b}}_{N+1}\|$). Applying lemma 6.9 then yields

$$\frac{(2N)!}{(2N^N)N!} \|\tilde{\mathbf{b}}_1\|^N \|\tilde{\mathbf{b}}_{N+1}\|^N \approx q^N$$

Stirling's formula allows us to conclude that $\|\tilde{\mathbf{b}}_{N+1}\| \approx \frac{qe}{2\|\tilde{\mathbf{b}}_1\|}$. Therefore, the value $\max(\|\tilde{\mathbf{b}}_1\|, \|\tilde{\mathbf{b}}_{N+1}\|)$ is minimized for $\|\mathbf{b}_1\| \approx \sqrt{\frac{qe}{2}} \approx 1.1658\sqrt{q}$. This matches closely the experimental results of Figure 6.2. For the key-generation algorithm to terminate faster, we take a slightly larger value, and chose $\|\mathbf{b}_1\| \approx 1.17\sqrt{q}$.

6.5.2 Optimal Bases for the Hybrid and Peikert's Samplers

The goal of this section is to explain why our Hybrid sampler performs better than Peikert's sampler over NTRU lattices.

We first recall properties of the singular norm. For two matrices $\mathbf{X}$ and $\mathbf{Y}$, and their vertical concatenation $\{\mathbf{X}, \mathbf{Y}\}$, we have:

$$\max(s_1(\mathbf{X})^2, s_1(\mathbf{Y})^2) \leqslant s_1(\{\mathbf{X}, \mathbf{Y}\})^2 \leqslant s_1(\mathbf{X})^2 + s_1(\mathbf{Y})^2. \tag{6.5}$$

The first inequality is an equality if and only if $\mathbf{X}$ and $\mathbf{Y}$ span orthogonal spaces. We also have euclidean sub-additivity: $s_1(\mathbf{X} + \mathbf{Y})^2 \leqslant s_1(\mathbf{X})^2 + s_1(\mathbf{Y})^2$.

We also recall that we write the NTRU basis as $\mathbf{B} = \{\mathbf{f}, \mathbf{F}\}$, where

$$\mathbf{f} = (g, -f) \text{ and } \mathbf{F} = (G, -F) = r\mathbf{f} + \tilde{\mathbf{F}}$$

for some $r \in \overline{\mathbb{K}}$ and $\tilde{\mathbf{F}}$ orthogonal to $\mathbf{f}$. The key generation process of NTRU guarantees that r has coefficients in $[-1/2, 1/2]$, and we modelize them as uniformly random in this range. Using our Hybrid sampler gives a quality of $s_1(\{\mathbf{f}, \tilde{\mathbf{F}}\}) = \max(s_1(\mathbf{f}), s_1(\tilde{\mathbf{F}}))$, against a quality of $s_1(\{\mathbf{f}, \mathbf{F}\})$ for Peikert's.

We now give three heuristics (two new and one already tested) that we will use to estimate the qualities of the samplers:

1. For $\mathbf{x} \in \{\mathbf{f}, \mathbf{F}, \tilde{\mathbf{F}}\}$, $s_1(\mathbf{x}) \approx \alpha\sqrt{\log N}\|\mathbf{f}\|$ for some constant α. To justify this, we observe that if $y \in \overline{\mathbb{K}}$'s coefficients are Gaussian (as in our experiments), then its embeddings $\sigma_i(y)$ are Gaussian as well. The singular $s_1(y)$ is the maximum absolute value of these embeddings $s_1(y) = \max_\sigma |\sigma_i(y)|$, which is expected, according to order statistics to be $\approx \alpha\sqrt{\log N}\|y\|$ for some constant α. Assuming $\mathbf{F}$, $\mathbf{f}$ and $\tilde{\mathbf{F}}$ behave like y, this gives the heuristic.

2. $\|\tilde{\mathbf{F}}\| \approx \frac{qe}{2\|\mathbf{f}\|}$. This heuristic is at the end of Section 6.5.1, and we also provide experimental confirmation in Section 6.4.1 and Figure 6.2.

3. $\|r\mathbf{f}\| \approx \sqrt{\frac{N}{12}}\|\mathbf{f}\|$. The heuristic can be found in [HHGP+03].

The quality of the Hybrid sampler is now discussed in points 1 and 2, and the one of Peikert's sampler in points 3 and 4.

1. Using heuristic 2, $\max(\|\mathbf{f}\|, \|\tilde{\mathbf{F}}\|)$ is minimized when $\|\mathbf{f}\| \approx \|\tilde{\mathbf{F}}\| \approx \sqrt{\frac{qe}{2}}$.

2. For the Hybrid sampler, the quality verifies:

$$s_1(\{\mathbf{f}, \tilde{\mathbf{F}}\}) = \max(s_1(\mathbf{f}), s_1(\tilde{\mathbf{F}})) \approx \alpha\sqrt{\log N}\max(\|\mathbf{f}\|, \|\tilde{\mathbf{F}}\|)$$

   where the equality is coming from the orthogonality of $\mathbf{f}$ and $\mathbf{F}$. How small we can expect $s_1(\{\mathbf{f}, \tilde{\mathbf{F}}\})$ to be then comes from point 1.

3. We can assume that $\max(\|\mathbf{f}\|^2, \|\mathbf{F}\|^2) = \|\mathbf{F}\|^2 \approx \|\mathbf{f}\|^2 + \|\mathbf{F}\|^2$ since

$$\|\mathbf{F}\|^2 = \|\tilde{\mathbf{F}}\|^2 + \|r\mathbf{f}\|^2 \approx \frac{q^2e^2}{4\|\mathbf{f}\|^2} + \frac{N}{12}\|\mathbf{f}\|^2$$

   where the approximation comes from heuristics 2 and 3. $\|\mathbf{F}\|$ is expected to be minimal for $\|\mathbf{f}\|^2 \approx qe\sqrt{\frac{3}{N}}$, which yields $\|\mathbf{F}\| \approx \sqrt{\frac{qe\sqrt{N}}{2\sqrt{3}}}$.

4. Using equation 6.5 in conjunction with heuristic 1 and point 3, we approximate the quality of Peikert's sampler:

$$s_1(\{\mathbf{f}, \mathbf{F}\}) \approx \alpha\sqrt{\log N}\|\mathbf{F}\|$$

We can now estimate the ratio between the best qualities obtained for a given dimension using the Hybrid and Peikert's samplers:

$$\frac{\sigma_{\texttt{Peikert}}}{\sigma_{\texttt{Hybrid}}} = \frac{s_1(\{\mathbf{f}, \mathbf{F}\})}{s_1(\{\mathbf{f}, \tilde{\mathbf{F}}\})} \approx \frac{\min_{\mathbf{f}} \|\mathbf{F}\|}{\min_{\mathbf{f}} \max(\|\mathbf{f}\|, \|\tilde{\mathbf{F}}\|)} \approx \left(\frac{N}{3}\right)^{1/4}$$

For $N = 512$, this ratio is about 3.6, which is close to the ratio of 3.2 that we observed in our experiments when optimizing both samplers.

However, if the samplers are run on the same lattice, the difference can be much higher; in particular, if $\|\mathbf{f}\| \gg \|\tilde{\mathbf{F}}\|$ then from the heuristic we can expect Peikert's sampler to have a quality worse than the Hybrid by a factor $\sqrt{N/12}$, but in practice this factor is even bigger, around $\sqrt{N/4}$ (see Figure 6.5).

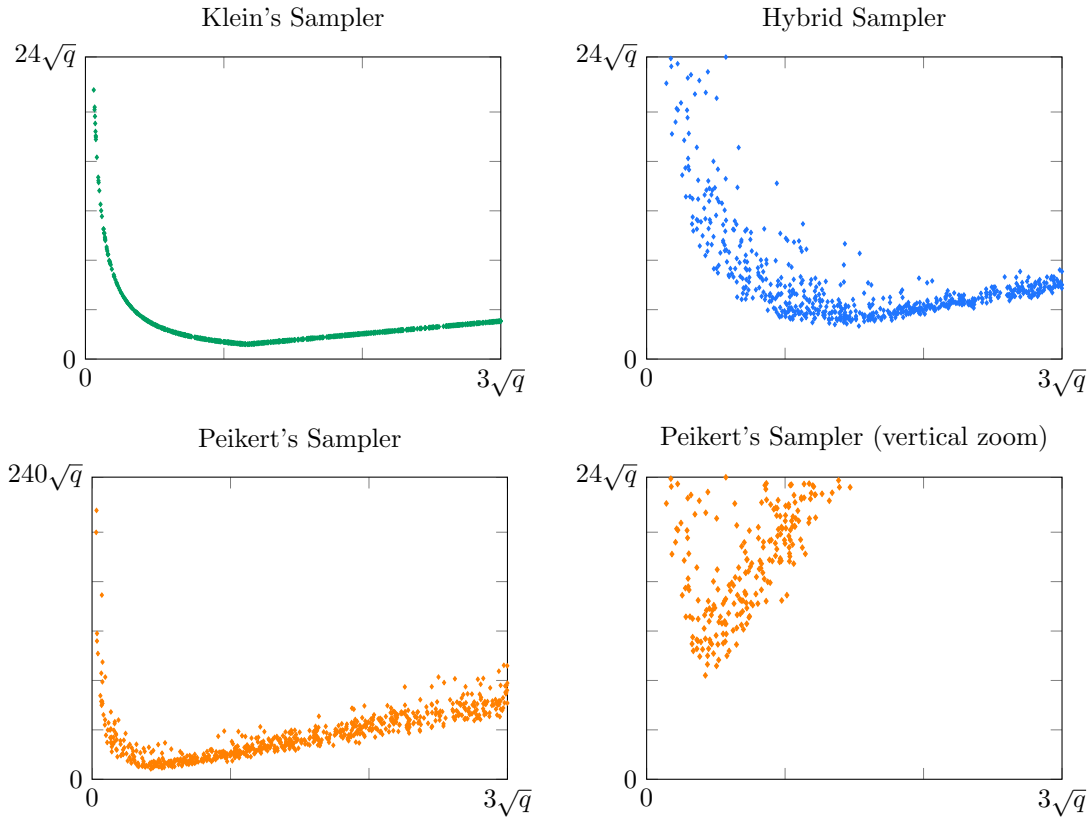


Figure 6.5: Comparing the qualities of Klein's/Hybrid/Peikert's samplers for a NTRU basis of dimension $2N = 1024$ and modulus $q = 2^{20}$, with $\|(f, g)\|$ (in abscissa) ranging in $(0, 3\sqrt{q})$.

6.6 Conclusion

We have provided an comparison of our hybrid sampler with the existing ones on NTRU lattices, which are the current most compact constructions. Klein's and Peikert's sampler both have their advantages, and on the lattices we experimented on, our sampler has the same complexity as Peikert's sampler, and is closer to Klein's sampler in terms of security. However, its running time still has to be assessed in practice. Additionally, in future works it would be interesting to evaluate our sampler in the context of the construction of Micciancio and Peikert [MP12].

Chapter 7

Fast Fourier Orthogonalization

"To control this ring seems wise to me. But how, Loge, can I learn the art of forging this gem?"

Wotan — Richard Wagner, *Das Rheingold*

7.1 Introduction

The nearest plane algorithm [Bab85, Bab86], which we studied extensively in this thesis in its deterministic and randomized forms (notably Section 2.2, Chapters 4 and 5), is a central algorithm over lattices. It allows, using a quadratic number of arithmetic operations, to find a relatively close point in a lattice to an arbitrary target. It is a core subroutine of LLL [LLL82], and can be used for error correction over analogical noisy channels. It also found applications in lattice-based cryptography as a decryption algorithm, and a randomized variant (called discrete Gaussian sampling) [Kle00, GPV08] provides secure trapdoor functions based on lattice problems. This leads to cryptosystems (identity-based and attribute-based encryption) with fine-grained access control [CHKP10, MP12, Boy13, GVW13].

When it comes to using this algorithm for practical cryptography, its quadratic cost is rather prohibitive, considering the lattices at hand have dimensions ranging in the hundreds. For efficiency purposes, many cryptosystems (such as [HPS98, LMPR08, LPR10] to name a few) chose to rely on lattices with some algebraic structure, improving time and memory requirements by a factor quasi-linear in the dimension. This is sometimes referred as lattice-based cryptography in the *ring-setting*. The core of this optimization is the Fast-Fourier Transform (FFT) [CT65, GS66, HJB84, Nus12] allowing fast multiplication of polynomials. But this improvement doesn't apply in the case of the nearest plane algorithm or its randomized variant [Kle00, GPV08]: Gram-Schmidt orthogonalization *seems* to break the algebraic structure.

To circumvent this issue, Peikert [Pei10] proposed to switch to a simpler algorithm, the so-called round-off algorithm [Len82, Bab85, Bab86], and using a perturbation technique, properly randomized it to obtain a secure discrete Gaussian sampler. This simpler algorithm is compatible with FFT-based acceleration techniques, but this comes at another price: the *quality* of the solution is affected. In short, instead of reducing the target to the cuboid associated with $\tilde{\mathbf{B}}$, this algorithm instead provides an output in the parallelepiped associated with $\mathbf{B}$: the fundamental domain has been skewed. This is pictured in Figure 2.1.

Quality in the ring setting. For this discussion, let us focus on a simple case: let $\mathbf{B} \in \mathbb{R}^{d \times d}$ be a *circulant* matrix of first row vector $\mathbf{b}_1 = (f_0, \ldots, f_{d-1})$. Algebraically, it is

Algorithm	Nearest plane	Round-off	Klein	Peikert
Quality	$\sqrt{\frac{1}{12}\sum_i \|\tilde{\mathbf{b}}_i\|^2}$	$\sqrt{\frac{d}{12}} \cdot \|\mathbf{b}_1\|$	$\sqrt{d}\,\|\mathbf{b}_1\|\eta_\epsilon(\mathbb{Z})$	$\sqrt{d}\,s_1(\mathbf{B})\eta_\epsilon(\mathbb{Z})$

Table 7.1: Average output length for random target ($\mathbf{B}$ circulant of first row $\mathbf{b}_1$).

the matrix associated to the multiplication by the element $f = \sum f_i \cdot x^i$ of the *circular convolution ring* $\mathcal{R} = \mathbb{R}[x]/(x^d - 1)$. The typical case encountered in cryptography deals with *cyclotomic rings* instead—and involves matrices with several blocks of this form—we will address this point later.

The question is to quantify the loss of quality when switching from the slow nearest plane to the fast round-off algorithm and their respective randomized variants, Klein's sampler and Peikert's sampler. We measure this as the average euclidean length of its output: the lower the better. This is summarized in Table 7.1. The factor $s_1(\mathbf{B})$ denotes the largest singular value of $\mathbf{B}$, which is also its spectral norm: $s_1(\mathbf{B}) = \max \|\mathbf{xB}\|/\|\mathbf{x}\|$. The factor η_ϵ is called the smoothing parameter [MR07], and some effort has been done to tackle it [DLP14, BLP+13, BLL+15], but the details are out of the scope of this chapter. For our matter, note that we have the inequalities:

$$\sqrt{\textstyle\sum_i \|\tilde{\mathbf{b}}_i\|^2} \leqslant \sqrt{d} \cdot \|\mathbf{b}_1\| \leqslant s_1(\mathbf{B}),$$

and equalities holds if and only if the matrix $\mathbf{B}$ is orthogonal. We call orthogonality defect the ratio $\delta(\mathbf{B}) = s_1(\mathbf{B})/\sqrt{d} \cdot \|\mathbf{b}_1\|$—alternatively one can consider $\delta'(\mathbf{B}) = \sqrt{d}\|\mathbf{b}_1\|/\sqrt{\sum_i \|\tilde{\mathbf{b}}_i\|^2}$.

The theory of random matrices (see [Ver10, Thm. 5.39]) predicts that the orthogonality defect $\delta(\mathbf{B})$ should tend to 1 as the dimension d grows, if the entries of $\mathbf{B}$ are centered and independent. But should it be the case for circulant matrices ? A simple experiment (see Fig. 7.1) shows it does not seem likely!

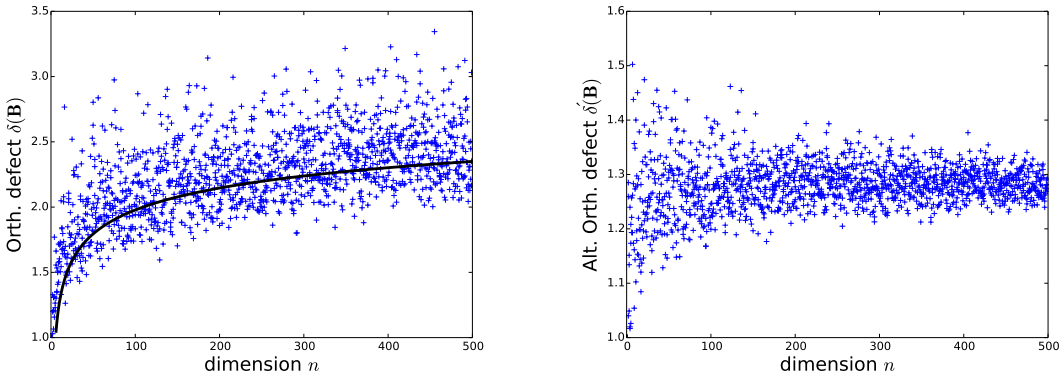


Figure 7.1: Experimental orthogonality defect of circulant matrices, and prediction (7.1)

This phenomenon is easily explained by switching to an algebraic point of view. Indeed, the singular values of a circulant matrix $\mathbf{B}$ are exactly the magnitudes of the FFT coefficients of f: the FFT diagonalizes the matrix $\mathbf{B}$, and this is the very reason why FFT allows fast multiplication in those rings. Also, the FFT is a scaled isometry: if the coefficients of f are independent Gaussians of variance σ^2, then the FFT coefficients

$\hat{f}_i = \sum_j f_j \zeta^{ij}$ have both their real and imaginary parts being independent Gaussians as well with variance $d\sigma^2/2$—except for $\hat{f}_0$. Their magnitudes $|\hat{f}_i|$ follow a Rayleigh distribution of parameter $\varsigma = \sqrt{d\sigma^2}$, whose density over $[0, \infty)$ is given by $\frac{x}{\varsigma^2} \exp\left(\frac{-x^2}{2\varsigma^2}\right)$. Out of $d/2$ (half may be ignored as conjugates of the other half) such samples, we expect the largest one to be as large as $\approx \varsigma\sqrt{\log d/2}$. Hence

$$\delta(\mathbf{B}) \approx \sqrt{\log(d/2)}, \tag{7.1}$$

and this closely matches our experiment.

Contribution. In this work, we discover chimeric algorithms, obtained by crossing Cooley-Tukey's [CT65] Fast-Fourier Transform algorithm together with an orthogonalization and nearest plane algorithms (not exactly Gram-Schmidt orthogonalization, but rather LDL decomposition). Precisely, we show that the nearest plane algorithm can be performed using quasi-linear arithmetic operations for matrices with the appropriate algebraic structures, coming from the d-th circular convolution ring when d is composed of small factors. The precomputed orthogonalization can also be done using quasi-linear many operations and produces the LDL decomposition in a special compact format, requiring $O(d \log d)$ complex numbers.

At the core of this technique is the realization that the circulant representation of matrices is not the appropriate one. Instead, switching to (mixed-)digit-reversal-order allows to represent the matrix $\mathbf{L}$ of the Gram-Schmidt decomposition in a compact inductive format, namely a tree of structured matrices, that get smaller and smaller as one reaches the leaves. Once this hidden structure is unveiled (Theorem 7.14), the algorithmic implications follow quite naturally. As a demonstration of the simplicity of those algorithms, we also propose an implementation in `python` in the power-of-two case. The code is strikingly short.

Most of the material of this chapter is very well known, but the authors are unaware of any work proposing such a combination. We chose to go very thoroughly through the details of both orthogonalization and Fast-Fourier Transform, to observe how nicely those two operations fit together.

Gaussian Sampling. We will only present our acceleration for the (deterministic) nearest plane algorithm. Generalizing our technique to the randomized algorithm of Klein is just a matter of replacing each call to the rounding function $\lfloor \cdot \rceil$, by an appropriately randomized rounding function—following a discrete Gaussian distribution over the integers.

Impact for lattice-based cryptography. The structure of matrices found for lattice-based cryptography follow from cyclotomic rings $\mathcal{F}_d = \mathbb{R}[x]/(\phi_d(x))$, and often d is chosen as power of 2. In fact, our algorithm can be generalized to handle such a case, because there is a ring morphism from $\mathcal{F}_d$ to $\mathcal{R}_d$ that is also an isometry. We merely chose the circulant-convolution ring as it makes our exposition significantly simpler.

Our work makes the nearest plane algorithm reach the same time complexity as the round-off algorithm for circulant matrices. The same phenomenon occurs to their randomized variants, the Gaussian samplers.

For samplers, the factor $\sqrt{\log(d/2)}$ to be saved may seem small, but it turns out that in practice the security of a cryptosystem is extremely sensitive to the quality of the Gaussian

sampler. Chapter 6 illustrated this phenomenon: for the same parameters, the signature scheme studied would have about 128 bits of security using Peikert's sampler, against up to 192 bits of security using Klein's sampler, and the hybrid sampler we proposed in Chapter 5 would reach 160 bits of security. In this study case, the seemingly anecdotal factor corresponded to 32 bits of security.

Related work. The first Gaussian sampler is due to Klein [Kle00], as a tool to solve the bounded-distance-decoding problem. Its use as a secure trapdoor algorithm was discovered and analyzed in the seminal work of Gentry, Peikert and Vaikuntanathan [GPV08], opening new horizons for lattice-based cryptography. In [LP15] it is showed how to decrease the pre-computation from cubic to quadratic in the ring setting, and how to store this pre-computation compactly without slowing the nearest plane algorithm. Alternatively, a FFT-compatible Gaussian Sampler was proposed by Peikert [Pei10] with quasilinear complexity but decreased quality. Optimization of the trapdoor basis generation (with a security reduction to a worst-case lattice problem [Ajt96, Reg05], unlike the trapdoors of [HPS98]) can be found in [AP11, MP12]. The question of fixed-point/floating-point precision was studied in [DN12a, Duc13], and optimized subroutines for sampling in 1-dimensional lattices were developed in [DN12a, DDLL13, Kar13, Duc13, BCG+14, RVV14, DG14, Lep14]. Improvements on the smoothing parameter using statistical tools can be found in [DLP14, BLL+15], and a dedicated algorithm was developed in [BLP+13].

Complexity. All the complexities are expressed in terms of number of arithmetic operation over the reals. Determining the required precision for a floating-point approximation and therefore the bit-complexity of our algorithm is left for future works as this issue seems rather orthogonal to our improvements. As a starting point in that direction, one might remark that the asymptotic relative error growth of the FFT of a degree-d polynomial is $O(\log d)$ in the worst case and $O(\sqrt{\log d})$ in the average case [GS66, Sch96]. As our algorithms are structurally very close to the FFT, it would be interesting to see if they benefit from the same error growth.

Open Problems. On the theoretical side, a interesting problem would be to apply this trick to the LLL [LLL82] algorithm, even if it weakens a bit the notion of reduction achieved. On the practical side, it would of course be nice to see a secure and fast implementation of lattice trapdoors based on this algorithm, combining the technique of [BLP+13, BLL+15] to tackle the smoothing parameter. This would have a wide impact, since many advanced cryptosystems as attribute-based encryption schemes [CHKP10, MP12, Boy13, GVW13] become rather easy to implement once this primitive is provided.

Organization. The chapter is organized as follows. First, Section 7.2 introduces the mathematical tools that we will use through this chapter. Section 7.3 shows that for matrices over convolution rings, the LDL decomposition can be expressed in a compact, factorized form, and gives a "Fast Fourier" algorithm for computing it in this form. This compact LDL decomposition is further exploited in Section 7.4, which presents a nearest plane algorithm that also has a "Fast Fourier" structure. Section 7.5 extends all our previous results from convolution rings to cyclotomic rings, by reducing the latter to the former. As a conclusion, Section 7.6 demonstrates the practical feasibility of our algorithms by presenting python implementations of them in the case where d is a power

of two. Appendix 7.A contains some proofs that were cut from the main body of the present chapter.

7.2 Preliminaries

We recall that most of the notation conventions are stated in pages 11 to 19.

7.2.1 The Convolution Ring $\mathcal{R}_d$

Definition 7.1. *For any $d \in \mathbb{N}^\star$, we note $\mathcal{R}_d$ the ring $\mathbb{R}[x]/(x^d - 1)$, also known as circular convolution ring, or simply convolution ring.*

When d is highly composite, elementary operations in $\mathcal{R}_d$ can be performed in time $O(d \log d)$ using the Fast Fourier transform [CT65].

We equip the ring $\mathcal{R}_d$ with a conjugation operation as well as an inner product, giving it an inner product space structure over $\mathbb{R}$. The definitions that we give also encompass other types of rings that will be used in later sections of this chapter.

Definition 7.2. *Let $h \in \mathbb{Q}[x]$ be a monic polynomial with distinct roots over $\mathbb{C}$, $\mathcal{R} \triangleq \mathbb{R}[x]/(h(x))$ and a, b be arbitrary elements of $\mathcal{R}$.*

- *We note $a^\star$ and call conjugate of a the unique element of $\mathcal{R}$ such that for any root ζ of h, $a^\star(\zeta) = \overline{a(\zeta)}$, where $\bar{\cdot}$ is the usual complex conjugation over $\mathbb{C}$.*
- *The inner product over $\mathcal{R}$ is defined by $\langle a, b \rangle \triangleq \sum_{h(\zeta)=0} a(\zeta) \cdot \overline{b(\zeta)}$, and the associated norm is $\|a\| \triangleq \sqrt{\langle a, a \rangle}$.*

One can check that if $a(x) = \sum_{d \in \mathbb{Z}_d} a_i x^i \in \mathcal{R}_d$, then

$$a^\star(x) = a(1/x) \bmod (x^d - 1) = \sum_{i \in \mathbb{Z}_d} a_i x^{d-i}$$

We extend the conjugation to matrices: if $\mathbf{B} = (b_{ij})_{i,j} \in \mathcal{R}^{n \times m}$, then the conjugate transpose of $\mathbf{B}$ is noted $\mathbf{B}^\star \in \mathcal{R}^{m \times n}$ and is the transpose of the coefficient-wise conjugation of $\mathbf{B}$.

While the inner product $\langle \cdot, \cdot \rangle$ (resp. the associated norm $\| \cdot \|$) is not to be mistaken with the canonical coefficient-wise dot product $\langle \cdot, \cdot \rangle_2$ (resp. the associated norm $\| \cdot \|_2$), they are closely related. One can easily check that for any $f = \sum_{0 \leqslant i < d} f_i x^i \in \mathcal{R}_d$, the vector $(f(\zeta))_{\{\zeta^d = 1\}}$ can be obtained from the coefficients' vector $(f_i)_{0 \leqslant i < d}$ by multiplying it by the Vandermonde matrix $\mathbf{V}_d = (\zeta_d^{ij})_{0 \leqslant i,j < d}$. $\mathbf{V}_d$ verifies $\mathbf{V}_d \mathbf{V}_d^\star = d \cdot \mathbf{I}_d$ and as an immediate consequence: $\langle f, g \rangle = d \cdot \langle f, g \rangle_2$.

Definition 7.3. *Let $m \geqslant n$ and $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}_d^{n \times m}$. We say that $\mathbf{B}$ is full-rank (or is a basis) if for any linear combination $\sum_{1 \leqslant i \leqslant n} a_i \mathbf{b}_i$ with $a_i \in \mathcal{R}_d$, we have the equivalence $(\sum_i a_i \mathbf{b}_i = 0) \iff (\forall i, a_i = 0)$.*

We note that since $\mathcal{R}_d$ is not an integral domain, a set formed of a single nonzero vector is not necessarily full-rank.

7.2.2 Coefficient Vectors and Circulant Matrices

Definition 7.4. *We define the coefficient vector* $c : \mathcal{R}_d^m \to \mathbb{R}^{dm}$ *and the circulant matrix* $\mathcal{C} : \mathcal{R}_d^{n\times m} \to \mathbb{R}^{dn\times dm}$ *as follows. For any* $a = \sum_{i\in\mathbb{Z}_d} a_i x^i \in \mathcal{R}_d$ *where each* $a_i \in \mathbb{R}$*:*

1. *The coefficients' vector of* a *is* $c(a) = (a_0, ..., a_{d-1}) \in \mathbb{R}^d$.
2. *The circulant matrix of* a *is*

$$\mathcal{C}(a) \triangleq \begin{bmatrix} a_0 & a_1 & \ddots & a_{d-1} \\ a_{d-1} & a_0 & \ddots & a_{d-2} \\ \ddots & \ddots & \ddots & \ddots \\ a_1 & a_2 & \dots & a_0 \end{bmatrix} = \begin{bmatrix} c(a) \\ c(xa) \\ \vdots \\ c(x^{d-1}a) \end{bmatrix} \in \mathbb{R}^{d\times d}.$$

3. c *and* $\mathcal{C}$ *generalize to vectors and matrices in a coefficient-wise manner.*

We give a few properties which are either folklore or trivial to verify.

Proposition 7.5. *The coefficients' vector and the circulant matrix verify the following properties:*
1. $\mathcal{C}$ *is a ring isomorphism onto its image. In particular* $\mathcal{C}(a)\mathcal{C}(b) = \mathcal{C}(ab)$.
2. $c(a)\mathcal{C}(b) = c(ab)$.
3. $\mathcal{C}(a)^\star = \mathcal{C}(a^\star)$.

7.2.3 Vectorize and Matrixify Operators

In this section, we introduce the "vectorize" and "matrixify" functions. Informally, they "break" elements of a convolution ring $\mathcal{R}_d$ into many elements of a smaller ring $\mathcal{R}_{d'}$, with $d'|d$. From a matricial point of view, they can also be seen as permuting rows and columns of a circulant matrix to turn it into a concatenation of smaller circulant matrices.

Definition 7.6. *Let* $d \in \mathbb{N}^\star$ *be a product of* h *(not necessarily distinct) primes. We note* $\mathrm{gpd}(d)$ *the greatest proper divisor of* d*. When clear from context, we also note* h *the number of prime divisors of* d *(counted with multiplicity),* $d_h \triangleq d$ *and for* $i \in [\![1, h]\!]$*,* $d_{i-1} \triangleq d_i/\mathrm{gpd}(d_i)$ *and* $k_i \triangleq d_i/d_{i-1}$*, so that* $1 = d_0|d_1|...|d_h = d$ *and* $\prod_{j\leqslant i} k_j = d_i$.

The d_i's defined in Definition 7.6 form a tower of proper divisors of d. For any composite d, there exist multiple towers of proper divisors: per example , $1|6$, $1|2|6$ and $1|3|6$ for $d = 6$. In this thesis, each time we mention a tower of proper divisors of d it will refer to the unique one induced by Definition 7.6.

Definition 7.7. *Let* $d, d' \in \mathbb{N}^\star$ *such that* $d'|d$*. We define the vectorization* $\mathsf{V}_{d\backslash d'} : \mathcal{R}_d^{n\times m} \to \mathcal{R}_d^{n\times m(d/d')}$ *inductively as follows:*

1. *Let* $k = d/\mathrm{gpd}(d)$*. For* $d' = \mathrm{gpd}(d)$ *and a single element* $a \in \mathcal{R}_d$*,* $a = \sum_{0\leqslant i<k} x^i a_i(x^k)$ *where* $a_i \in \mathcal{R}_{d'}$ *for each* i*. Then*

$$\mathsf{V}_{d\backslash d'}(a) \triangleq (a_0, ..., a_{k-1}) \in \mathcal{R}_{d'}^k$$

In other words, $\mathsf{V}_{d\backslash d'}(a)$ *is the row vector whose coefficients are the* $(a_i)_{0\leqslant i<k}$.

2. *For a vector* $\mathbf{v} \in \mathcal{R}_d^m$ *or a matrix* $\mathbf{B} \in \mathcal{R}_d^{n\times m}$, $\mathsf{V}_{d\backslash d'}(\mathbf{v}) \in \mathcal{R}_d^{(d/d')m}$ *and* $\mathsf{V}_{d\backslash d'}(\mathbf{B}) \in \mathcal{R}_d^{n\times(d/d')m}$ *are the componentwise applications of* $\mathsf{V}_{d\backslash d'}$.

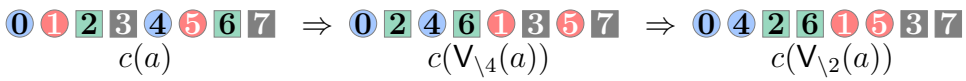
3. *For* $d''|d'|d$ *and any element* $a \in \mathcal{R}_d$,

$$\mathsf{V}_{d\backslash d''}(a) \triangleq \mathsf{V}_{d'\backslash d''} \circ \mathsf{V}_{d\backslash d'}(a) \in \mathcal{R}_{d''}^{d/d''}$$

When d *is clear from context, we simply note* $\mathsf{V}_{d\backslash d'} = \mathsf{V}_{\backslash d'}$.

Interpretation.

In practice, an element $a \in \mathcal{R}_d$ is represented by a vector of d elements corresponding to the d coefficients of a. In this context, the vectorization operation simply permutes coefficients. As highlighted by Figure 7.2, when $d = 2^h$ is a power of two, $\mathsf{V}_{d\backslash 1}$ permutes the coefficients according to the bit-reversal order,[1] which appears in the radix-2 Fast Fourier transform (FFT). More generally, one can show that for an arbitrary d, $\mathsf{V}_{d\backslash 1}$ permutes the coefficient according to the more general mixed-radix digit reversal order, which appears in the mixed-radix Cooley-Tukey FFT.



The vectorization operation "breaks" the element $0 + x + ... + 7x^7$ of $\mathcal{R}_8$ into two elements $0 + 2x + 4x^2 + 6x^3$ and $1 + 3x + 5x^2 + 7x^3$ of $\mathcal{R}_8$, then into four elements of $\mathcal{R}_2$.

Figure 7.2: Vectorizations of $a = 0 + x + 2x^2 + \dots 7x^7$

Similarly, one can define an operation which we call "matrixification". Like the vectorization breaks any element of $\mathcal{R}_d$ into a vector, the matrixification breaks it into a matrix.

Definition 7.8. *Following the notations of Definition 7.7, we define the matrixification* $\mathsf{M}_{d\backslash d'} : \mathcal{R}_d^{n\times m} \to \mathcal{R}_{d'}^{n(d/d')\times m(d/d')}$ *as follows:*

1. *For* $d' = \mathrm{gpd}(d), k = d/d'$ *and a single element* $a = \sum_{0\leqslant i<k} x^i a_i(x^k)$ *where each* $a_i \in \mathcal{R}_{d'}$*:*

$$\mathsf{M}_{d\backslash d'}(a) \triangleq \begin{bmatrix} a_0 & a_1 & \ddots & a_{k-1} \\ xa_{k-1} & a_0 & \ddots & a_{k-2} \\ \ddots & \ddots & \ddots & \ddots \\ xa_1 & xa_2 & \dots & a_0 \end{bmatrix} = \begin{bmatrix} \mathsf{V}_{d\backslash d'}(a) \\ \mathsf{V}_{d\backslash d'}(x^k a) \\ \vdots \\ \mathsf{V}_{d\backslash d'}(x^{(d'-1)k} a) \end{bmatrix} \in \mathcal{R}_{d'}^{nk\times mk}$$

In particular, if d *is prime, then* $\mathsf{M}_{d\backslash 1}(a) \in \mathbb{R}^{d\times d}$ *is exactly the circulant matrix* $\mathcal{C}(a)$.

2. *For a vector* $\mathbf{v} \in \mathcal{R}_d^m$ *or a matrix* $\mathbf{B} \in \mathcal{R}_d^{n\times m}$, $\mathsf{M}_{d\backslash d'}(\mathbf{v}) \in \mathcal{R}_d^{(d/d'')\times(d/d')m}$ *and* $\mathsf{M}_{d\backslash d'}(\mathbf{B}) \in \mathcal{R}_d^{(d/d')n\times(d/d')m}$ *are the componentwise applications of* $\mathsf{M}_{d\backslash d'}$.

3. *For any element* $a \in \mathcal{R}_d$,

$$\mathsf{M}_{d\backslash d''}(a) \triangleq \mathsf{M}_{d'\backslash d''} \circ \mathsf{M}_{d\backslash d'}(a) \in \mathcal{R}_{d''}^{(d/d'')\times(d/d'')}$$

[1] https://oeis.org/A030109

When d is clear from context, we simply note $\mathsf{M}_{d\backslash d'} = \mathsf{M}_{\backslash d'}$.

Proposition 7.9. *Let $d \in \mathbb{N}^\star$, a, b (resp. $\mathbf{a}, \mathbf{b}$, resp. $\mathbf{A}, \mathbf{B}$) be arbitrary scalars (resp. vectors, resp. matrices) over $\mathcal{R}_d$, and $d'|d$. For concision, we note $\mathsf{V} \triangleq \mathsf{V}_{d\backslash d'}$ and $\mathsf{M} \triangleq \mathsf{M}_{d\backslash d'}$. The vectorization and matrixification verify the following properties:*

1. *The matrixification is an algebra isomorphism onto its image, and in particular $\mathsf{M}(\mathbf{A} \cdot \mathbf{B}) = \mathsf{M}(\mathbf{A}) \cdot \mathsf{M}(\mathbf{B})$*
2. *The vectorization is a vector space isomorphism onto its image.*
3. *$\mathsf{V}(ab) = \mathsf{V}(a) \cdot \mathsf{M}(b)$*
4. *The vectorization is an isometry for the canonical coefficient-wise scalar product $\langle \cdot, \cdot \rangle_2$:*

$$\langle \mathsf{V}(\mathbf{a}), \mathsf{V}(\mathbf{b}) \rangle_2 = \langle \mathbf{a}, \mathbf{b} \rangle_2$$

5. *$\mathbf{B}$ is full-rank if and only if $\mathsf{M}(\mathbf{B})$ is full-rank.*

Since the proofs are rather straightforward to check from the definitions, we leave them in Appendix 7.A.1.

Interpretation.

In lattice-based cryptography, cryptosystems using ring lattices rely on the hardness of problems on lattices over convolution rings.[2] In this setting, the basis is constituted of the rows of (concatenations of) circulant matrices. A prevalent example are the NTRU lattices generated by bases of the form

$$\left[\begin{array}{c|c} \mathcal{C}(f) & \mathcal{C}(g) \\ \hline \mathcal{C}(F) & \mathcal{C}(G) \end{array}\right].$$

If we identify an element $a \in \mathcal{R}_d$ with its circulant matrix $\mathcal{C}(a)$, all the matrixification does is permute rows and columns of $\mathcal{C}(a)$. Permuting the rows clearly leaves invariant the lattice generated by the matrix. On the other hand, permuting the columns changes the lattice generated, but since it preserves the scalar product, the geometry of the lattice isn't affected.

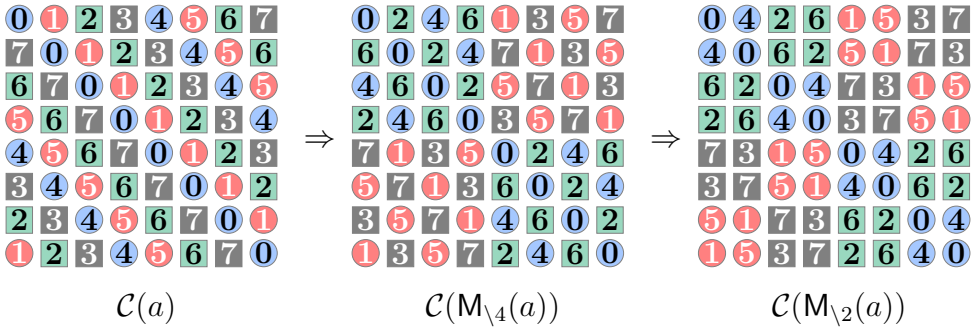


Figure 7.3: Matrixifications of $a = 0 + x + 2x^2 + \ldots 7x^7$

[2] In reality, the underlying rings used are more often cyclotomic rings. However, we reduce the cyclotomic ring case to the convolution ring case in the ulterior Section 7.5.

Computing the Vectorization and Matrixification in the Fourier Domain.

An interesting observation about the "vectorize" and "matrixify" operators that we defined is that they can be computed very efficiently when an element $a \in \mathcal{R}_d$ is represented by its coefficients but also when it is represented in the *Fourier domain*. In the first case, it is obvious that since these operations permutes coefficients of a, they can both be performed in time $O(d)$.

If a is represented in FFT form, computing its vectorization and matrixification over $\mathcal{R}_{d'}$ in FFT form can naively be done in time $O(d \log d)$ by computing its reverse FFT, permuting its coefficients and computing d/d' FFT's over $R_{d'}$. However, we observe that it can be done faster, since it is essentially equivalent to a step of the original Fast Fourier transform. This is formalized in Lemma 7.10, a reformulation of a simple lemma that is at the heart of Cooley-Tukey's FFT.

Lemma 7.10 ([CT65], adapted)**.** *Let $d \geqslant 2, d' = \mathrm{gpd}(d), k = d/d'$ and $a \in \mathcal{R}_d$ be uniquely written as $a = \sum_{0 \leqslant i < k} x^i a_i(x^k)$ where each $a_i \in \mathcal{R}_{d'}$. One can compute the FFT of a from the FFT's of the a_i's (and reciprocally) in time $O(dk)$. In particular, $\mathsf{V}_{d\backslash d'}(a)$, its inverse and $\mathsf{M}^{-1}_{d\backslash d'}((a_i)_i)$ (resp. $\mathsf{M}_{d\backslash d'}(a)$) can be computed in FFT form in time $O(kd)$ (resp. $O(k^2 d)$).*

Proof. In [CT65, equations 7 and 8], Cooley and Tukey show that one can switch from the FFT of a to the FFT of the a_i's (and conversely) in time $O(kd)$. Since the a_i's are the coefficients of $\mathsf{V}_{d\backslash d'}(a)$ and $\mathsf{M}_{d\backslash d'}(a)$, the result follows. □

Lemma 7.10 allows us to gain a factor $O(\log d)$ when computing the vectorization and matrixification of a, compared to a naive approach. In Sections 7.3 and 7.4, we will define algorithms which heavily rely on these operators, and will therefore benefit from this speedup as well.

7.2.4 The LDL Decomposition

In this section and the next one, $\mathcal{R} \triangleq \mathcal{R}_d$ for some $d \geqslant 1$. We recall the LDL decomposition, which is widespread inside and outside the scope of lattice-based cryptography. First, we define the notion of lower triangular unit matrix, which will be prevalent in the rest this chapter.

Definition 7.11. *We say that a matrix $\mathbf{L} \in \mathcal{R}^{n \times n}$ is lower triangular unit (or LTU) if it is lower triangular and has only 1's on its diagonal.*

The LDL decomposition is complementary to the Gram-Schmidt decomposition. It writes any symmetric definite positive matrix – which is symmetric matrix $\mathbf{G} \in \mathcal{R}^{n \times n}$ such that for any $\mathbf{x} \neq \mathbf{0}$, $\mathbf{x}\mathbf{G}\mathbf{x}^\star >_{\mathcal{R}} \mathbf{0}$ – as a product $\mathbf{L}\mathbf{D}\mathbf{L}^\star$, where $\mathbf{L} \in \mathcal{R}^{n \times n}$ is lower triangular with 1's on the diagonal, and $\mathbf{D} \in \mathcal{R}^{n \times n}$ is diagonal.

To properly define the notion of definite positiveness, we would need to define a semi-order $\geqslant_{\mathcal{R}}$ over elements of $\mathcal{R}$.[3] To avoid it, we resort to the notion of full-rank Gram matrix, which is essentially equivalent to definite positiveness.

Definition 7.12. *We say that a matrix $\mathbf{G} \in \mathcal{R}^{n \times n}$ is full-rank Gram (or FRG) if it is full-rank and there exists $m \geqslant n$ and $\mathbf{B} \in \mathcal{R}^{n \times m}$ such that $\mathbf{G} = \mathbf{B}\mathbf{B}^\star$.*

[3] The usual semi-order over $\mathcal{R} = \mathbb{R}[x]/(h(x))$ is: $(a \geqslant_{\mathcal{R}} b) \Leftrightarrow (|a(\zeta)| \geqslant |b(\zeta)|$ for any root ζ of $h)$.

One can show that a matrix is FRG if and only if it is symmetric definite positive. However, the former notion requires less definitions and is much simpler to manipulate from an algorithmic viewpoint, so we will rely on it instead of the latter.

Algorithm 7.1 $\mathsf{LDL}_{\mathcal{R}}(\mathbf{G})$

Require: A full-rank Gram matrix $\mathbf{G} = (G_{ij}) \in \mathcal{R}^{n \times n}$ defined over a ring $\mathcal{R}$
Ensure: The decomposition $\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^\star$ over $\mathcal{R}$, where $\mathbf{L}$ is LTU and $\mathbf{D}$ is diagonal
1: $\mathbf{L}, \mathbf{D} \leftarrow \mathbf{0}^{n \times n}$
2: **for** i from 1 to n **do**
3: $\quad L_{ii} \leftarrow 1$
4: $\quad D_i \leftarrow G_{ii} - \sum_{j<i} L_{ij} L_{ij}^\star D_j$
5: $\quad$ **for** j from 1 to $i-1$ **do**
6: $\quad\quad L_{ij} \leftarrow \frac{1}{D_j}\left(G_{ij} - \sum_{k<j} L_{ik} L_{jk}^\star D_k\right)$
7: $\quad$ **end for**
8: **end for**
9: **return** $((L_{ij}), \mathrm{diag}(D_i))$

We now explicit the relation between both decompositions. For a basis $\mathbf{B}$, there exists a unique Gram-Schmidt decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$ and for a FRG matrix $\mathbf{G}$, there exists a unique LDL decomposition $\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^\star$. If $\mathbf{G} = \mathbf{B}\mathbf{B}^\star$, then $\mathbf{G}$ is FRG and one can check that $\mathbf{G} = \mathbf{L} \cdot (\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\star) \cdot \mathbf{L}^\star$ is a valid LDL decomposition of $\mathbf{G}$. As both decompositions are unique, the matrices $\mathbf{L}$ in both decompositions are actually the same.

7.2.5 Babai's Nearest Plane Algorithm and Klein's Sampler

We give here an alternative definition of the nearest plane algorithm. Unlike the one we considered until now, all the internal computations are done not on a vector $\mathbf{c}$, but on the vector $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$, which represents $\mathbf{c}$ as a combination of vectors of $\mathbf{B}$. This slight difference will allow, in Section 7.4, to get a speed-up on a recursive variant of this algorithm.

Algorithm 7.2 $\mathsf{NearestPlane}(\mathbf{B}, \mathbf{L}, \mathbf{D}, \mathbf{c})$

Require: The decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$ over $\mathbb{R}$, a vector $\mathbf{c} \in \mathrm{Span}\,\mathbb{R}(\mathbf{B})$
Ensure: A vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\mathbf{c} - \mathbf{v} \in \mathcal{P}(\tilde{\mathbf{B}})$
1: $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
2: **for** $j = n, ..., 1$ **do**
3: $\quad \bar{t_j} \leftarrow t_j + \sum_{i>j} (t_i - z_i) L_{ij}$
4: $\quad z_j \leftarrow \lfloor \bar{t_j} \rceil$
5: **end for**
6: **return** $\mathbf{v} \leftarrow \mathbf{z} \cdot \mathbf{B}$

Proposition 7.13. *Algorithm 7.2 outputs* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\mathbf{c} - \mathbf{v} \in \mathcal{P}(\tilde{\mathbf{B}})$.

The proof of Proposition 7.13 is standard and deferred to Appendix 7.A.2.

7.3 Fast Fourier LDL Decomposition

7.3.1 A Compact Representation for the LDL Decomposition

Theorem 7.14. *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of d. Let $\mathbf{b} \in \mathcal{R}_d^m$ be a full-rank vector. There exists a Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{b})$ as follows:*

$$\mathsf{M}_{d\backslash 1}(\mathbf{b}) = \left(\prod_{i=0}^{h-1} \mathsf{M}_{d_i\backslash 1}(\mathbf{L}_i)\right) \cdot \mathbf{B}_0$$

where $\mathbf{B}_0 \in \mathbb{R}^{d\times dm}$ is orthogonal and each $\mathbf{L}_i \in \mathcal{R}_{d_i}^{(d/d_i)\times(d/d_i)}$ is a block-diagonal matrix whose (d/d_{i+1}) diagonal blocks are lower triangular unit (LTU) matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$.[4]

As a toy example, the matrix $\mathbf{L}$ of the Gram-Schmidt decomposition of $\mathsf{M}_{4\backslash 1}(\mathbf{a})$ for an element $\mathbf{a} \in \mathcal{R}_4^m$ would look like this:

$$\left[\begin{array}{cc|cc} 1 & & & \\ & 1 & & \\ \hline a & b & 1 & \\ b & a & & 1 \end{array}\right] \cdot \left[\begin{array}{cc|cc} 1 & & & \\ c & 1 & & \\ \hline & & 1 & \\ & & d & 1 \end{array}\right].$$

Proof. If d is prime, the theorem is trivial. We suppose that d is composite and that the theorem is true for any $\mathcal{R}_i$ with $i < d$. By Proposition 7.9, item 5, the matrix $\mathbf{B}_{h-1} \triangleq \mathsf{M}_{d\backslash d_{h-1}}(\mathbf{b})$ is full-rank too. We can therefore decompose it as $\mathbf{B}_{h-1} = \mathbf{L}_{h-1}\tilde{\mathbf{B}}$, where $\mathbf{L}_{h-1} \in \mathcal{R}_{d_{h-1}}^{k_d\times k_d}, \tilde{\mathbf{B}} \in \mathcal{R}_{d_{h-1}}^{k_d\times mk_d}$ and $k_d \triangleq d/\operatorname{gpd}(d)$. $\mathbf{L}_{h-1}$ is LTU and $\tilde{\mathbf{B}}$ is orthogonal. Noting $\tilde{\mathbf{B}} = [\mathbf{b}_1, ..., \mathbf{b}_{k_d}]$, each vector $\mathbf{b}_j$ is full-rank and orthogonal to the other $\mathbf{b}_{j'}$'s. By inductive hypothesis, they can be decomposed as follows:

$$\forall j \in [\![1, n]\!], \mathsf{M}_{d_{h-1}\backslash 1}(\mathbf{b}_j) = \left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i\backslash 1}(\mathbf{L}_{i,j})\right) \tilde{\mathbf{B}}_j \tag{7.2}$$

Where each $\tilde{\mathbf{B}}_j \in \mathbb{R}^{d_{h-1}\times md_{h-1}}$ is full-rank orthogonal and for $i < h-1$, each $\mathbf{L}_{i,j} \in \mathcal{R}_{d_i}^{(d_{h-1}/d_i)\times(d_{h-1}/d_i)}$ is a block-diagonal matrix whose (d_{h-1}/d_{i+1}) diagonal blocks are lower triangular unit (LTU) matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$. For concision, we now note $\mathsf{M} \triangleq \mathsf{M}_{d_{h-1}\backslash 1}$ and $\mathsf{V} \triangleq \mathsf{V}_{d_{h-1}\backslash 1}$. We have:

$$\begin{aligned}
\mathsf{M}_{d\backslash 1}(\mathbf{b}) &= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \mathsf{M}(\tilde{\mathbf{B}}_{h-1}) \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \mathsf{M}[\mathbf{b}_1, ..., \mathbf{b}_{k_d}] \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot [\mathsf{M}(\mathbf{b}_1), ..., \mathsf{M}(\mathbf{b}_{k_d})] \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \operatorname{diag}\left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i\backslash 1}(\mathbf{L}_{i,j})\right) [\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_{k_d}] \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i\backslash 1}(\mathbf{L}_i)\right) \mathbf{B}_0 \\
&= \left(\prod_{i=0}^{h-1} \mathsf{M}_{d_i\backslash 1}(\mathbf{L}_i)\right) \mathbf{B}_0
\end{aligned}$$

The first equality simply uses the fact that M is a ring homomorphism. The second and third ones are immediate from the definitions. The fourth one uses the inductive

[4] The indexed products are to be read $\prod_{i=0}^{k} \alpha_i = \alpha_k\alpha_{k-1}...\alpha_0$.

hypothesis (equation 7.2) on each $\mathbf{b}_j$. In the fifth equality, we take $\mathbf{L}_i \triangleq \text{diag}(\mathbf{L}_{i,1}, ..., \mathbf{L}_{i,k_d})$ and $\mathbf{B}_0 \triangleq [\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_{k_d}]$ and just need to check that they are as stated by the theorem:

- Since each $\mathbf{L}_{i,j}$ is block diagonal with (d_{h-1}/d_{i+1}) LTU diagonal blocks, $\mathbf{L}_i$ is block diagonal with $k_d(d_{h-1}/d_{i+1}) = (d/d_{i+1})$ LTU diagonal blocks.
- We also need to show that $\mathbf{B}_0$ is orthogonal. Each submatrix $\tilde{\mathbf{B}}_j$ of $\mathbf{B}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{b}_j)$ by induction hypothesis. Therefore, for two distinct rows $\mathbf{u}, \mathbf{v}$ of $\mathbf{B}_0$:
  - If they belong to the same submatrix $\tilde{\mathbf{B}}_j$, they are orthogonal by induction hypothesis.
  - Suppose they belong to different submatrices: $\mathbf{u} \in \tilde{\mathbf{B}}_j, \mathbf{v} \in \tilde{\mathbf{B}}_\ell$ and $j \neq \ell$. Then $\mathbf{u}$ (resp. $\mathbf{v}$) is a linear combination of rows of $\mathsf{M}(\mathbf{b}_j)$ (resp. $\mathsf{M}(\mathbf{b}_\ell)$): $\mathbf{v} = \mathbf{a}_j \cdot \mathsf{M}(\mathbf{b}_j)$ and $\mathbf{v} = \mathbf{a}_\ell \cdot \mathsf{M}(\mathbf{b}_j)$ for some $\mathbf{a}_j, \mathbf{a}_\ell$ in $\mathbb{R}^{d_{h-1}}$. Noting $a_j = \mathsf{V}^{-1}(\mathbf{a}_j)$ and $a_\ell = \mathsf{V}^{-1}(\mathbf{a}_\ell)$, we have:
$$\begin{aligned} \langle \mathbf{u}, \mathbf{v} \rangle &= \langle \mathsf{V}(a_j)\mathsf{M}(\mathbf{b}_j), \mathsf{V}(a_\ell)\mathsf{M}(\mathbf{b}_\ell) \rangle \\ &= \langle \mathsf{V}(a_j \mathbf{b}_j), \mathsf{V}(a_\ell \mathbf{b}_\ell) \rangle \\ &= \langle a_j \mathbf{b}_j, a_\ell \mathbf{b}_\ell \rangle = 0 \end{aligned}$$
    Where the second equality comes from Proposition 7.9, item 3, the third one from the fact that V is a scaled isometry and the fourth one from the fact that $\mathbf{b}_j, \mathbf{b}_\ell$ are orthogonal.

  Therefore $\mathbf{B}_0$ is orthogonal.

□

The theorem we stated gives the Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{b})$ for a vector $\mathbf{b} \in \mathcal{R}_d^m$, but can be easily generalized from a vector $\mathbf{b}$ to a matrix $\mathbf{B}$, and also yields a compact LDL decomposition.

Corollary 7.15. *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of d. Let $\mathbf{B} \in \mathcal{R}_d^{n \times m}$ be a full-rank matrix. There exist $h+1$ matrices $(\mathbf{L}_i)_{0 \leqslant i \leqslant h}$ where:*

- *$\mathbf{L}_h \in \mathcal{R}_d^{n \times n}$ is LTU.*
- *For each $i < h$, $\mathbf{L}_i \in \mathcal{R}_{d_i}^{n(d/d_i) \times n(d/d_i)}$ is a block-diagonal matrix whose $n(d/d_{i+1})$ diagonal blocks are LTU matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i) \times (d_{i+1}/d_i)}$.*

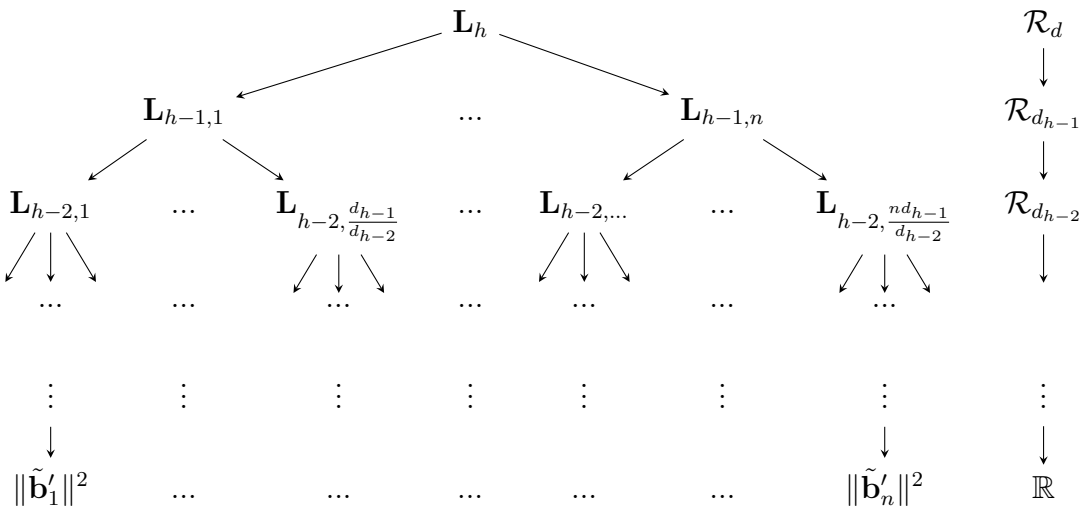
Such that, if we note $\mathbf{L} = \left(\prod_{i=0}^{h} \mathsf{M}_{d_i \backslash 1}(\mathbf{L}_i)\right)$ and $\mathbf{B}_0 \triangleq \mathbf{L}^{-1} \cdot \mathsf{M}_{d\backslash 1}(\mathbf{B})$:

1. *The Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{B})$ is $\mathsf{M}_{d\backslash 1}(\mathbf{B}) = \mathbf{L} \cdot \mathbf{B}_0$.*
2. *The LDL decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{B}\mathbf{B}^\star)$ is $\mathsf{M}_{d\backslash 1}(\mathbf{B}\mathbf{B}^\star) = \mathbf{L} \cdot (\mathbf{B}_0 \mathbf{B}_0^t) \cdot \mathbf{L}^t$.*

Proof. We have $\mathbf{B} = \mathbf{L}_h \mathbf{B}'$, where $\mathbf{L}_h$ is given by either the Gram-Schmidt or LDL decomposition algorithm. $\mathbf{B}' = \{\mathbf{b}'_1, ..., \mathbf{b}'_n\}$ is orthogonal. Applying Theorem 7.14 to each row vector $\mathbf{b}'_j$ of $\mathbf{B}'$ yields n decompositions $(\mathbf{L}_{i,j})_{0 \leqslant i < h}$ and n orthogonal matrices $\tilde{\mathbf{B}}_j$, each spanning the same space as $\mathbf{B}_j \triangleq \mathsf{M}_{d\backslash 1}(\mathbf{b}'_j)$. Taking $\mathbf{L}_i \triangleq \text{diag}(\mathbf{L}_{i,j})$ and $\mathbf{B}_0 \triangleq [\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_{0,n}]$ yields the Gram-Schmidt decomposition.

The LDL decomposition is then given "for free" by the correspondence between it and the Gram-Schmidt decomposition, and indeed, one can check that since $\mathbf{B}_0$ is orthogonal, $(\mathbf{B}_0 \mathbf{B}_0^t)$ is diagonal. □

Theorem 7.14 and Corollary 7.15 state that for any full-rank matrix $\mathbf{B} \in \mathcal{R}^{n \times m}$, the $\mathbf{L}$ matrix in its Gram-Schmidt and LDL decompositions can be represented in a factorized form, where each of the factors $\mathbf{L}_i$ is a sparse, block-diagonal matrix. Figure 7.4 synthesizes this factorization.



$\mathbf{L}_h \in \mathcal{R}_d^{n \times n}$ is a LTU matrix, and for each $i < h$, every $\mathbf{L}_{i,j}$ is a lower triangular unit matrix in $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$. Each of the matrices $\mathbf{L}_i$ mentioned in Theorem 7.14 is the block-diagonal matrix whose blocks are the $(\mathbf{L}_{i,j})_{1 \leqslant j \leqslant nd/d_{i+1}}$.

Figure 7.4: Tree $\mathfrak{L}$ of precomputed matrices $\mathbf{L}_{i,j}$ such that $\mathbf{L} = \prod_i \mathsf{M}_{d_i \backslash 1}(\mathrm{diag}_j(\mathbf{L}_{i,j}))$

7.3.2 A Fast Algorithm for the Compact LDL Decomposition

Theorem 7.14 and Corollary 7.15 are constructive: more precisely, their proofs give an algorithm to compute the compact factorized form of $\mathbf{L}$ quickly. Algorithm 7.3 performs the compact LDL decomposition in the form of the tree given in Figure 7.4.

Algorithm 7.3 $\mathsf{ffLDL}_{\mathcal{R}_d}(\mathbf{G})$

Require: A full-rank Gram matrix $\mathbf{G} \in \mathcal{R}_d^{n \times n}$
Ensure: The compact LDL decomposition of $\mathbf{G}$
1: **if** $d = 1$ **then**
2: **return** $(\mathbf{G}, [\,])$
3: **end if**
4: $(\mathbf{L}, \mathbf{D}) \leftarrow \mathsf{LDL}_{\mathcal{R}_d}(\mathbf{G})$
5: **for** $i = 1, ..., n$ **do**
6: $\mathfrak{L}_i \leftarrow \mathsf{ffLDL}_{\mathcal{R}_{\mathrm{gpd}(d)}}(\mathsf{M}_{d \backslash \mathrm{gpd}(d)}(\mathbf{D}_{ii}))$
7: **end for**
8: **return** $(\mathbf{L}, (\mathfrak{L}_i)_{1 \leqslant i \leqslant n})$

Algorithm 7.3 computes a "Fast Fourier LDL", instead of the "Fast Fourier Gram-Schmidt" hinted at in Theorem 7.14 and Corollary 7.15. The reason why we favor this

approach is because it allows a complexity gain. This gain already occurs in the classic versions of the aforementioned algorithms.

As a simple example, consider the $\mathbf{L}$ in the Gram-Schmidt decomposition of $\mathbf{B} \in \mathcal{R}_d^{2\times m}$, which is exactly the $\mathbf{L}$ in the LDL decomposition of $\mathbf{BB}^\star \in \mathcal{R}_d^{2\times 2}$. Computing it with the LDL algorithm will be $O(m)$ times faster than with the Gram-Schmidt algorithm. The same phenomenon happens with their recursive variants.

Lemma 7.16. *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of d, and for $i \in [\![1, h]\!]$, let $k_i \triangleq d_i/d_{i-1}$. Let $\mathbf{G} \in \mathcal{R}_d^{n\times n}$ be a full-rank Gram matrix. Then Algorithm 7.3 computes the LDL decomposition tree of $\mathbf{G}$ in FFT form in time*

$$O(n^2 d \log d) + O(n^3 d) + O(nd) \sum_{1 \leqslant i \leqslant h} k_i^2$$

In particular, if all the k_i are bounded by a small constant k, then the complexity of Algorithm 7.3 is upper bounded by $O(n^3 d + n^2 d \log d)$.

Proof. Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 7.3 over a matrix $\mathbf{G} \in \mathbb{R}_d^{k\times k}$. We have the following recursion formula:

$$\mathrm{C}(n, d) = O(n^2 d \log d) + O(n^3 d) + O(dk_h^2) + n\mathrm{C}(k_h, d_{h-1}) \tag{7.3}$$

Where the first term corresponds to computing the FFT of the n^2 coefficients of $\mathbf{G}$, and the second term to performing $(\mathbf{L}, \mathbf{D}) \leftarrow \mathsf{LDL}_{\mathcal{R}_d}(\mathbf{G})$ in FFT form. For each $i \in [\![1, n]\!]$, we know from Lemma 7.10 that $\mathsf{M}_{d\setminus \mathrm{gpd}(d)}(\mathbf{D}_{ii})$ can be computed in time $O(dk_h^2)$, hence the third term. The last one is for the n recursive calls to itself. We then have

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1 \leqslant i \leqslant h} \tfrac{d}{d_i} O(d_{i-1} k_i^3) + \tfrac{d}{d_1} \mathrm{C}(k_1, d_0) \\ &= O(d) \sum_{1 \leqslant i \leqslant h} k_i^2 \end{aligned} \tag{7.4}$$

Where the first equality is shown by induction using equation 7.3, *except* the first term $O(n^2 d \log d)$ which is no longer relevant since we are already in the Fourier domain. Combining equations 7.3 and 7.4, we conclude that the complexity of the total algorithm is

$$\begin{aligned} \mathrm{C}(n, d) &= O(n^2 d \log d) + O(n^3 d) + n\mathrm{C}(k_h, d_{h-1}) \\ &= O(n^2 d \log d) + O(n^3 d) + O(nd) \sum_{1 \leqslant i \leqslant h} k_i^2 \end{aligned}$$

□

7.4 Fast Fourier Nearest Plane

In this section, we show how to exploit further the compact form of the LDL decomposition to have a Fast Fourier variant of the nearest plane algorithm. It outputs vectors of the same quality as its classical, iterative counterpart, but runs $\tilde{O}(d)$ times faster.

Definition 7.17. *We note $\mathcal{Z}_d$ the ring $\mathbb{Z}[x]/(x^d - 1)$ of elements of $\mathcal{R}_d$ with integer coefficients.*

Algorithm 7.4 $\mathsf{ffNearestPlane}_{\mathcal{R}_d}(\mathbf{t}, \mathfrak{L})$

Require: $\mathbf{t} \in \mathcal{R}_d^n$, a precomputed tree $\mathfrak{L}$, (implicitly) a matrix $\mathbf{B} \in \mathcal{R}_d^{n \times m}$ such that $\mathfrak{L}$ is the compact LDL decomposition tree of $\mathbf{B}\mathbf{B}^\star$
Ensure: $\mathbf{z} \in \mathcal{Z}_d^n$ such that
1: **if** $\mathbf{t}$ is a 1-dimensional vector in $\mathbb{R}$ **then**
2: **return** $\lfloor \mathbf{t} \rceil$
3: **end if**
4: $\mathbf{L} \leftarrow \mathfrak{L}.\mathsf{Node}()$
5: **for** $j = n, ..., 1$ **do**
6: $\overline{\mathbf{t}}_j \leftarrow \mathbf{t}_j + \sum_{i>j}(\mathbf{t}_i - \mathbf{z}_i)L_{ij}$
7: $\mathbf{z}_j \leftarrow \mathsf{V}^{-1}_{d\backslash \mathrm{gpd}(d)} \left[\mathsf{ffNearestPlane}_{\mathcal{R}_{\mathrm{gpd}(d)}}(\mathsf{V}_{d\backslash \mathrm{gpd}(d)}(\overline{\mathbf{t}}_j), \mathfrak{L}.\mathsf{Child}(j))\right]$
8: **end for**
9: **return** $\mathbf{z}$

Lemma 7.18. *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$ be its GSO in $\mathcal{R}$. The vectors $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_n)$ and $\overline{\mathbf{t}} = (\overline{\mathbf{t}}_1, ..., \overline{\mathbf{t}}_n)$ in Algorithm 7.4 verify*

$$(\mathbf{z} - \mathbf{t}) \cdot \mathbf{B} = (\mathbf{z} - \overline{\mathbf{t}}) \cdot \tilde{\mathbf{B}}$$

Proof. We recall that for each $i \in [\![1, n]\!]$, $\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j<i} L_{ij}\tilde{\mathbf{b}}_j$. We have:

$$\begin{array}{rcl} (\mathbf{z} - \overline{\mathbf{t}}) \cdot \tilde{\mathbf{B}} & = & \sum_{j=1...n}(\mathbf{z}_j - \overline{\mathbf{t}}_j) \cdot \tilde{\mathbf{b}}_j \\ & = & \sum_{j=1...n}\left[(\mathbf{z}_j - \mathbf{t}_j) + \sum_{i>j}(\mathbf{z}_i - \mathbf{t}_i) \cdot L_{ij}\right] \cdot \tilde{\mathbf{b}}_j \\ & = & \sum_{1 \leqslant i \leqslant j \leqslant n}(\mathbf{z}_i - \mathbf{t}_i) \cdot L_{ij} \cdot \tilde{\mathbf{b}}_j \\ & = & \sum_{i=1...n}(\mathbf{z}_i - \mathbf{t}_i) \cdot \mathbf{b}_i \\ & = & (\mathbf{z} - \mathbf{t}) \cdot \mathbf{B} \end{array} \tag{7.5}$$

The first and last equalities are trivial, the second one replaces the $\overline{\mathbf{t}}_j$'s by their definitions, the third one simplifies the sum and the fourth one is another way of saying that $\mathbf{L} \cdot \tilde{\mathbf{B}} = \mathbf{B}$. □

Theorem 7.19. *Let $\mathsf{M} \triangleq \mathsf{M}_{d\backslash 1}$ and $\mathsf{V} \triangleq \mathsf{V}_{d\backslash 1}$. Algorithm 7.4 outputs $\mathbf{z} \in \mathcal{Z}_d^n$ such that $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}(\mathbf{B}_0)$, where $\mathbf{B}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{B})$.*

Proof. The result is trivially true if $n = d = 1$. We prove it in the general case. By definition, each $\mathfrak{L}.\mathsf{Child}(j)$ is the LDL decomposition tree of $\mathsf{M}_{d\backslash \mathrm{gpd}(d)}(\tilde{\mathbf{b}}_j)$. By induction hypothesis, we therefore know that $\mathsf{V}((\mathbf{z}_j - \overline{\mathbf{t}}_j)\tilde{\mathbf{b}}_j) \in \mathcal{P}(\tilde{\mathbf{B}}_j)$, where $\tilde{\mathbf{B}}_j$ is the orthogonalization of $\mathbf{B}_j \triangleq \mathsf{M}(\tilde{\mathbf{b}}_j)$. From Lemma 7.18, we have:

$$(\mathbf{z} - \mathbf{t})\mathbf{B} = \sum_{j=1...n}(\mathbf{z}_j - \overline{\mathbf{t}}_j) \cdot \tilde{\mathbf{b}}_j$$

so $\mathsf{V}((\mathbf{z}-\mathbf{t})\mathbf{B}) \in \mathcal{P}([\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n])$. Now, from the proof of Corollary 7.15, we know that $\mathbf{B}_0 = [\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n]$ is actually the orthogonalization of $\mathsf{M}(\mathbf{B})$, which concludes the proof. □

Equivalently, Theorem 7.19 states that Algorithm 7.4 outputs $\mathbf{z}$ such that $(\mathbf{z}-\mathbf{t})\mathbf{B} \in \mathsf{V}^{-1}\widetilde{\mathcal{P}(\mathsf{M}(\mathbf{B}))}$. This behavior is slightly different from the usual nearest plane algorithm. When instantiated for maximal efficiency from a quality point of view (therefore completely ignoring the ring structure), the latter outputs $\mathbf{z}$ such that $(\mathbf{z}-\mathbf{t})\mathbf{B} \in c^{-1}\widetilde{\mathcal{P}(\mathcal{C}(\mathbf{B}))}$.

Although $(\mathbf{z}-\mathbf{t})\mathbf{B}$ is in a different cuboid in each case, both algorithms are essentially identical in terms of quality. Ignoring the action of V^{-1}, c^{-1} (since they are isometries), the cuboids $\widetilde{\mathcal{P}(\mathsf{M}(\mathbf{B}))}$ and $\widetilde{\mathcal{P}(\mathcal{C}(\mathbf{B}))}$ predictably have the same volume. But more importantly, they are generated by nd orthogonal vectors, and one can easily show that the length of the longest one is the same in both cases. Therefore, Algorithm 7.4 is essentially as good as the nearest plane algorithm in terms of output quality. In addition, it can be run significantly faster over convolution ring lattices, as demonstrated in Lemma 7.20.

Lemma 7.20. *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of d, and for $i \in [\![1, h]\!]$, let $k_i \triangleq d_i/d_{i-1}$. Let $\mathbf{B} \in \mathcal{R}_d^{n\times m}$ and $\mathfrak{L}$ be its LDL decomposition tree. The complexity of Algorithm 7.4 is upper bounded by:*

$$O(nd\log d) + O(n^2 d) + O(nd)\sum_{1\leqslant i\leqslant h} k_i^2$$

In particular, if all the k_i are bounded by a small constant k, then the complexity of Algorithm 7.4 is upper bounded by $O(n^2 d + nd\log d)$.

Proof. Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 7.4 over input $\mathbf{t} \in \mathcal{R}_d^k$. We have this recursion formula:

$$\mathrm{C}(n, d) = O(nd\log d) + O(n^2 d) + O(ndk_h^2) + n\mathrm{C}(k_h, d_{h-1})$$

Where the first term corresponds to computing the FFT of the n coefficients of $\mathbf{t}$, the second term to performing computing the $\overline{\mathbf{t}}_j$'s (step 6) in FFT form, the third one to the n calls to $\mathsf{V}^{-1}_{d\setminus \mathrm{gpd}(d)}, \mathsf{M}_{d\setminus \mathrm{gpd}(d)}$ (see Lemma 7.10) and the fourth one to the n recursive call to itself. We have:

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1\leqslant i\leqslant h} \tfrac{d}{d_i} O(d_{i-1}k_i^3) + \tfrac{d}{d_1}\mathrm{C}(k_1, d_0) \\ &= O(d)\sum_{1\leqslant i\leqslant h} k_i^2 \end{aligned} \tag{7.6}$$

Where the equalities are obtained using the same reasoning as in the proof of Lemma 7.16. Similarly, we can then conclude that:

$$\mathrm{C}(n, d) = O(nd\log d) + O(n^2 d) + O(nd)\sum_{1\leqslant i\leqslant h} k_i^2$$

□

7.5 Extending the Results to Cyclotomic Rings

In this section we argue that our result also applies in the cyclotomic case. It turns out that all the previous arguments can be made more general. The required ingredients are the following:

1. A tower of unitary rings endowed with inner products onto $\mathbb{R}$.
2. For any rings $\mathcal{S}, \mathcal{T}$ of the tower, injective maps $\mathsf{M}' : \mathcal{S} \to \mathcal{T}^{k\times k}$ and $\mathsf{V}' : \mathcal{S} \to \mathcal{T}^{k}$, with $\mathcal{S}$ of rank d over $\mathbb{R}$, and $\mathcal{T}$ of rank d/k over $\mathbb{R}$.
3. M' is a ring morphism.
4. V' is a scaled linear isometry.
5. $\mathsf{V}'(ab) = \mathsf{M}'(a)\mathsf{V}'(b)$.
6. Computing V', V'^{-1}, M' and M'^{-1} takes time $O(dk)$.

It remains to prove the existence of such maps for towers of cyclotomic rings. We give explicit constructions in this section, using both our maps from the previous sections and a generic embedding from cyclotomic rings $\mathcal{F}_d$ to convolution rings $\mathcal{R}_d$.

7.5.1 Cyclotomic Rings

Definitions and properties of cyclotomic elements are given in Section 4.2.4. We introduce a few additional definitions.

Definition 7.21. *Let $d \in \mathbb{N}^\star$. We recall that for $d \in \mathbb{N}^\star$, ζ_d denotes an arbitrary primitive d-th root of unity in $\mathbb{C}$, $\Omega_d = \{\zeta_d^k | k \in \mathbb{Z}_d^\times\}$ is the set of primitive d-th roots of unity and ϕ_d is the d-th cyclotomic polynomial. We define the polynomial ψ_d as follows:*

$$\psi_d(x) = \prod_{\zeta^d=1, \zeta\notin\Omega_d} (x-\zeta) = \prod_{k\in(\mathbb{Z}_d\setminus\mathbb{Z}_d^\times)} (x-\zeta_d^k)$$

We also note $\mathcal{F}_d$ the cyclotomic ring $\mathbb{R}[x]/(\phi_d(x))$.

One can check that $\phi_d(x)\cdot\psi_d(x) = x^d - 1$.

7.5.2 Embedding the Ring $\mathcal{F}_d$ in the Ring $\mathcal{R}_d$

We now explicit an embedding of $\mathcal{F}_d$ into $\mathcal{R}_d$.

Definition 7.22. *Let e_d be the unique element in $\mathcal{R}_d$ such that $e_d = 1 \bmod \phi_d$ and $e_d = 0 \bmod \psi_d$. We define the embedding ι_d from $\mathcal{F}_d$ into $\mathcal{R}_d$ as follows:*

$$\begin{array}{rccc} \iota_d & : & \mathcal{F}_d & \to & \mathcal{R}_d \\ & & f & \mapsto & f\cdot e_d. \end{array}$$

When clear from context, we simply note $\iota = \iota_d$.

Equivalently, $\iota(f)$ is the only element in $\mathcal{R}_d$ verifying:

$$\iota(f)(\zeta) = \begin{cases} f(\zeta) & \text{if } \phi_d(\zeta) = 0 \\ 0 & \text{if } \psi_d(\zeta) = 0 \end{cases} \tag{7.7}$$

Proposition 7.23. *Let $d \in \mathbb{N}^\star$ and $\iota = \iota_d$. The embedding ι:*

1. is a ring isomorphism onto its image.

2. is an isometry : for any $f, g \in \mathcal{F}_d$, $\langle \iota(f), \iota(g) \rangle = \langle f, g \rangle$.

Proof. Let us first prove item 1. The element e_d is idempotent in $\iota(\mathcal{F}_d)$: $e_d^2 = e_d$. From this, one can easily show that ι is a ring homomorphism. In addition, for any element $g \in \iota(\mathcal{F}_d)$, $g \bmod \phi_d$ is the unique antecedent of g with respect to ι, so ι is bijective and $\iota^{-1}(g) = g \bmod \phi_d$, which proves the point 1. Items 2. and 3. follows from equation (7.7). □

Lemma 7.24. *Let $d \geqslant 2, d'|d, k = d/d'$ and $a \in \mathcal{R}_d$. Then*

$$(a \in \iota(\mathcal{F}_d)) \Leftrightarrow \mathsf{V}_{d\backslash d'}(a) \in \iota(\mathcal{F}_{d'})^k$$

For readability, the proof of Lemma 7.24 is left in Appendix 7.A.3.

7.5.3 Conclusion for Cyclotomic Rings

We now check that the 6 conditions stated at the beginning of Section 7.5 are verified. For $d'|d$, $\mathcal{F}_{d'}$ and $\mathcal{F}_{d'}$ are unitary rings endowed with the dot product defined in Definition 7.2, which gives the condition 1. The embeddings ι_d trivialize the construction of maps M' and V' from $\mathcal{F}_d$ to $\mathcal{F}_{d'}$:

$$\mathsf{V}' = \iota_{d'}^{-1} \circ \mathsf{V}_{d\backslash d'} \circ \iota_d \qquad \mathsf{M}' = \iota_{d'}^{-1} \circ \mathsf{M}_{d\backslash d'} \circ \iota_d.$$

This gives the condition 2. Lemma 7.24 allows to argue that the image of $\mathsf{V}_{d/d'} \circ \iota_d$ is in the definition domain of $\iota_{d'}^{-1}$: V' is well defined, and similarly for M'. Conditions 3 and 5 follow from the fact that $\iota_d, \iota_{d'}$ are ring morphisms and that similar properties hold for $\mathsf{M}_{d\backslash d'}$ and $\mathsf{V}_{d\backslash d'}$. Condition 4 is true because ι_d, $\mathsf{V}_{d\backslash d'}$ and $\iota_{d'}$ are isometries. Finally, condition 6 holds in the FFT representation, from Lemma 7.10 and the from the fact that ι in the Fourier domain simply consist of inserting some 0 at appropriate positions.

7.6 Implementation in Python

In this final section, we give the core of the python implementation of our algorithm when d is a power of 2. This complete, public-domain implementation can be found at:

https://github.com/lducas/ffo.py

It includes a verifier `verif.py`, that is based on the (slow) Gram-Schmidt algorithm. The file `ffo.py` is the full version of the simplified algorithms given below. The file `ffo_NaN.py` is a slightly more evolved version that also handles the non-full rank case. The file `cyclo.py` implement the embedding technique of Section 7.5 to apply our algorithms to the cyclotomic ring setting. The file `test.py` runs test in the rings $\mathcal{R}_{64}$ and $\mathcal{F}_{64}$.

Conventions. In `python.numpy`, the arithmetic operations `+,-,*,/` on arrays denotes coefficient-wise operations. The functions `fft` and its inverse `ifft` are built in. The symbol `j` denotes the imaginary unit.

```
# Modified extract of ffo.py
from numpy import *

# Inverse vectorize operation V^-1, i/o in fft format
def ffmerge(F1,F2):
    d = 2*len(F1)
    F = 0.j*zeros(d)     # Force F to complex float type
    w = exp(-2.j*pi / d)
    W = array([w**i for i in range(d/2)])
    F[:d/2] = F1 + W * F2
    F[d/2:] = F1 - W * F2
    return F

# Vectorize operation V, i/o in fft format
def ffsplit(F):
    d = len(F)
    winv = exp(2.j*pi / d)
    Winv = array([winv**i for i in range(d/2)])
    F1 = .5* (F[:d/2] + F[d/2:])
    F2 = .5* (F[:d/2] - F[d/2:]) * Winv
    return (F1,F2)

# ffLDL alg., i/o in fft format, outputs an L-Tree (sec 3.2)
def ffLDL(G):
    d = len(G)
    if d==1:
        return (G,[])
    (G1,G2) = ffsplit(G)
    L  = G2 / G1
    D1 = G1
    D2 = G1 - L * G1 * conjugate(L)
    return (L, [ffLDL(D1),ffLDL(D2)] )

# ffLQ, i/o in fft format, outputs an L-Tree (sec 3.2)
def ffLQ(f):
    F = fft(f)
    G = F*conjugate(F)
    T = ffLDL(G)
    return T

# ffBabai alg., i/o in base B, fft format
def ffBabai_aux(T,t):
    if len(t)==1:
        return array([round(t.real)])
    (t1,t2) = ffsplit(t)
    (L,[T1,T2]) = T
    z2  = ffBabai_aux(T2,t2)
    tb1 = t1 + (t2-z2) * conjugate(L)
    z1  = ffBabai_aux(T1,tb1)
    return ffmerge(z1,z2)

# ffBabai alg., i/o in canonical base, coef. format
def ffBabai(f,T,c):
    F = fft(f)
    t = fft(c) / F
    z = ffBabai_aux(T,t)
    return ifft(z * F)
```

7.A Proofs

This section contains the proofs that were cut from the main body of Chapter 7 in order to improve readability.

7.A.1 Proof of Proposition 7.9

Proof. We show the properties separately:

1. We first prove this statement for $d' = \mathrm{gpd}(d)$ and for elements $a, b \in \mathcal{R}_d$. All the requirements for showing that M is a homomorphism are trivial except for the fact that it is multiplicative. First, one can check from Definition 7.8 that $\mathsf{M}(ab) = \mathsf{M}(a) \cdot \mathsf{M}(b)$. Let $\mathbf{A} = (a_{ij}) \in \mathcal{R}_d^{n\times p}$ and $\mathbf{B} = (b_{ij}) \in \mathcal{R}_d^{p\times m}$. Since $\mathbf{AB} \triangleq (\sum_{1\leqslant k\leqslant p} a_{ik}b_{kj})_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}$, we have
$$\begin{aligned}\mathsf{M}(\mathbf{AB}) &= \mathsf{M}\left(\left(\textstyle\sum_{1\leqslant k\leqslant p} a_{ik}b_{kj}\right)_{i,j}\right)\\ &= \left(\textstyle\sum_{1\leqslant k\leqslant p} \mathsf{M}(a_{ik})\mathsf{M}(b_{kj})\right)_{i,j}\\ &= \mathsf{M}(\mathbf{A})\mathsf{M}(\mathbf{B})\end{aligned}$$
Multiplicity then seamlessly transfers to any $d''|d$:
$$\begin{aligned}\mathsf{M}_{d\setminus d''}(\mathbf{A}\cdot\mathbf{B}) &= \mathsf{M}_{d'\setminus d''} \circ \mathsf{M}_{d\setminus d'}(\mathbf{A}\cdot\mathbf{B})\\ &= \mathsf{M}_{d'\setminus d''}(\mathsf{M}_{d\setminus d'}(\mathbf{A})\cdot\mathsf{M}_{d\setminus d'}(\mathbf{B}))\\ &= \mathsf{M}_{d'\setminus d''} \circ \mathsf{M}_{d\setminus d'}(\mathbf{A})\cdot\mathsf{M}_{d'\setminus d''} \circ \mathsf{M}_{d\setminus d'}(\mathbf{B})\\ &= \mathsf{M}_{d\setminus d''}(\mathbf{A})\cdot\mathsf{M}_{d'\setminus d''}(\mathbf{B})\end{aligned}$$
To show injectivity, it suffices to see that if $d' = \mathrm{gpd}(d)$, then $(\mathsf{M}_{d\setminus d'}(a) = 0) \Leftrightarrow (a = 0)$. From the definition, this property seamlessly transfers to any $d'|d$ and any matrix $\mathbf{A}$.

2. This item is immediate from the definition.

3. It suffices to notice that for any a, $\mathsf{V}(a)$ is the first line of $\mathsf{M}(a)$. As M is a multiplicative homomorphism, the result follows.

4. It suffices to prove it for elements $a, b \in \mathcal{R}_d$ (instead of vectors) and for $d' = \mathrm{gpd}(d)$, the generalization to vectors and to arbitrary values of d' is then immediate. Let $a = \sum_i x^i a_i(x^{\mathrm{gpd}(d)})$, $b = \sum_i x^i b_i(x^{\mathrm{gpd}(d)})$, where $\forall i, a_i = \sum_{0\leqslant j<i} a_{i,j}x^j$ and $b_i = \sum_{0\leqslant j<i} b_{i,j}x^j$. Then
$$\langle a, b\rangle_2 \triangleq \sum_{i,j}\langle a_{i,j}, b_{i,j}\rangle_2 = \sum_i \langle a_i, b_i\rangle_2 \triangleq \langle \mathsf{V}(a), \mathsf{V}(b)\rangle_2$$

5. We have:
$$\begin{array}{ccccc}(\mathbf{B}\text{ FR}) & \Leftrightarrow & (\forall \mathbf{a}, \mathbf{aB} = 0 \text{ iff } \mathbf{a} = 0) & \Leftrightarrow & (\forall \mathbf{a}, \mathsf{V}(\mathbf{aB}) = 0 \text{ iff } \mathsf{V}(\mathbf{a}) = 0)\\ & & & & \Updownarrow\\ (\mathsf{M}(\mathbf{B})\text{ FR}) & \Leftrightarrow & (\forall \mathbf{a}', \mathbf{a}'\mathsf{M}(\mathbf{B}) = 0 \text{ iff } \mathbf{a}' = 0) & \Leftrightarrow & (\forall \mathbf{a}', \mathsf{V}(\mathbf{a})\mathsf{M}(\mathbf{B}) = 0 \text{ iff } \mathsf{V}(\mathbf{a}) = 0)\end{array}$$
The left equivalences are simply the definition, the middle ones uses the fact that V is a vector space isomorphism and the right one uses Proposition 7.9, item 3.

□

7.A.2 Proof of Proposition 7.13

Proof. Let $\mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^\star$ be the LDL decomposition of $\mathbf{B}\mathbf{B}^\star$. We have:

$$\begin{aligned}(\mathbf{v}-\mathbf{c}) \cdot \tilde{\mathbf{B}}^\star &= (\mathbf{z}-\mathbf{t})\mathbf{B} \cdot \tilde{\mathbf{B}}^\star \\ &= (\mathbf{z}-\mathbf{t})\mathbf{L}\mathbf{D}\end{aligned} \tag{7.8}$$

One can check that for any $j \in [\![1, n]\!]$,

$$((\mathbf{z}-\mathbf{t})\mathbf{L})_j = \sum_{i \geqslant j}(z_i - t_i)L_{ij} = \bar{t}_j - z_j \in \left[-\frac{1}{2}, \frac{1}{2}\right] \tag{7.9}$$

where the second (resp. third) equality comes from the way the $\bar{t}_j$'s (resp. z_j's) are computed in Algorithm 7.2. We note $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n$ the row vectors of $\tilde{\mathbf{B}}$. Combining equations 7.8 and 7.9 yields $|\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle| \leqslant \frac{1}{2}\|\tilde{\mathbf{b}}_j\|^2$. Since the vectors $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n$ are orthogonal and $(\mathbf{v}-\mathbf{c}) \in \mathrm{Span}(\tilde{\mathbf{B}})$, we can write

$$\begin{aligned}\mathbf{v}-\mathbf{c} &= \sum_{1 \leqslant j \leqslant n} \frac{\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j \\ &\in \sum_{1 \leqslant j \leqslant n} \left[-\tfrac{1}{2}, \tfrac{1}{2}\right] \tilde{\mathbf{b}}_j \\ &\in \mathcal{P}(\tilde{\mathbf{B}})\end{aligned}$$

Where the second equality comes from the fact that $|\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle| \leqslant \frac{1}{2}\|\tilde{\mathbf{b}}_j\|^2$, and the third one from the definition of $\mathcal{P}$. This concludes the proof. □

7.A.3 Proof of Lemma 7.24

Proof. We prove the lemma for $d' = \mathrm{gpd}(d)$, extension to the general case is straightforward. a can be uniquely written as $a = \sum_{0 \leqslant i < k} x^i a_i(x^k)$ where each $a_i \in \mathcal{R}_{d'}$. Let ζ_d be an arbitrary d-th primitive root of unity. We recall that $\Omega_d = \{\zeta_d^j | j \in \mathbb{Z}_d^\times\}$ and note $U_d \triangleq \{\zeta \in \mathbb{C} | \zeta^d = 1\} = \{\zeta_d^j | j \in \mathbb{Z}_d\}$. One can check that:

$$(\zeta \in U_d \backslash \Omega_d) \Leftrightarrow (\zeta^k \in U_{d'} \backslash \Omega_{d'}) \tag{7.10}$$

Which is immediate by writing $\zeta = \zeta_d^j$, with $j \in \mathbb{Z}_d \backslash \mathbb{Z}_d^\times$. We recall that evaluating a on each $\zeta_d^j \in U_d$ yields the linear system

$$a(\zeta_d^j) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_d^{kj}) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_{d'}^j) \tag{7.11}$$

As a step of the FFT (see Lemma 7.10), the system 7.11 is invertible. In addition, one can check in equation 7.10 that if $\zeta \in U_d \backslash \Omega_d$, then $a(\zeta)$ depends only of the $a_i(\zeta')$ for $\zeta' \in U_{d'} \backslash \Omega_{d'}$. Similarly, if $\zeta \in \Omega_d$, then $a(\zeta)$ depends only of the $a_i(\zeta')$ for $\zeta' \in \Omega_{d'}$. So the linear system can be separated in two independent systems. Noting $a(E) \triangleq \{a(e) | e \in E\}$:

$$\left[\begin{array}{c|c} a(\Omega_d) & a(U_d \backslash \Omega_d) \end{array}\right] = \left[\begin{array}{c|c} (a_i(\Omega_{d'}))_{0 \leqslant i < k} & (a_i(U_{d'} \backslash \Omega_{d'}))_{0 \leqslant i < k} \end{array}\right] \left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M}_2 \end{array}\right]$$

Since the whole system is invertible, both matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are invertible too. We can conclude that $a(U_{d'} \backslash \Omega_{d'}) = \mathbf{0}^{d-\varphi(d)}$ iff all the $a_i(U_{d'} \backslash \Omega_{d'})$'s are zero too. This is equivalent to saying that $a \in \iota(\mathcal{F}_d)$ iff $\forall i, a_i \in \iota(\mathcal{F}_{d'})$, which proves the lemma. □

Part III

Identity-Based Encryption

Overview

Identity-Based Cryptography (IBC) is a paradigm of cryptography where an users' public key can be directly derived from their public information, like emails or IP addresses. The cost of the Public Key Infrastructure is therefore greatly reduced, since there is no need to distribute the public keys. A drawback is that the private-key generation is assured by a Private-Key Generator (PKG) which has knowledge of all the private keys. The first example of IBC has been proposed to solve key management issues by Shamir [Sha84], who achieved identity-based signatures using the existing RSA function. However, identity-based encryption remained an open question for 16 years.
The first IBE schemes were concurrently introduced by Sakai, Ohgishi and Kasahara [SOK00], Boneh and Franklin [BF01] and Cocks [Coc01] and were based respectively on pairings and quadratic residues. This was followed by several IBE construction over pairing groups. Recently, [GPV08] have proposed a framework for realize identity-based encryption using lattice-based cryptography, but no efficient instantiation of this framework had been realized. This part of the thesis intend to fill this gap.

In Chapter 8, we propose an efficient identity-based encryption scheme over lattices. To accomplish this, we take the framework from [GPV08] and instantiate it over NTRU lattices. Incidentally, this IBE scheme is a natural extension from the signature scheme proposed in Chapter 6 – which is expected since this is already the case for the IBE and signature schemes of [GPV08] that they are derived from.
This allows us to reuse results from the previous chapters to optimize the parameters of the scheme. Chapter 3 allowed to sample narrower Gaussians while still be able to claim the same level of security and Chapter 6 provided distributions for the NTRU lattices that are nearly optimal for Gaussian sampling. In addition, we propose some tweaks to the scheme that in particular allow to use our KL divergence-based argument and divide the ciphertexts size by two.
To conclude, we implemented our IBE scheme and compared it to a reasonably fast implementation of the Boneh-Franklin [BF01] IBE scheme. While our scheme has bigger users' key and ciphertext sizes – respectively about 40 times and twice bigger –, its encryption and decryption operations are faster by *three orders of magnitude*. We believe this gap is so huge that it makes lattice-based IBEs a viable alternative to their pairing-based counterparts, especially for constrained device whose computational power is low.

The Chapter 9 gives a few reminders on identity-based encryption as well as a few applications. In addition, we propose a potential new application: botnets. These networks of infected computers are faced with unusual constraints, including an openly hostile environment. We believe that IBE – and in particular lattice-based IBE – may provide an answer for some of the challenges that botnets face.

The Chapter 8 develops ideas originally mentioned in ***Efficient Identity-Based Encryption over NTRU Lattices*** [DLP14], a joint work with Léo Ducas and Vadim Lyubashevsky that was published at ASIACRYPT. The Chapter 9 is independent work.

Chapter 8

Efficient Identity-Based Encryption over NTRU Lattices

8.1 Introduction

Recent improvements in efficiency have firmly established lattice-based cryptography as one of the leading candidates to replace number-theoretic cryptography after the eventual coming of quantum computing. There are currently lattice-based encryption [HPS98, LPR13a, LPR13b], identification [Lyu12b], and digital signature schemes [PDG14, DDLL13] that have run-times (both in software and in hardware), key sizes, and output lengths that are more or less on par with traditional number-theoretic schemes. But unfortunately, the extent of practical lattice-based cryptography stops here. While number-theoretic assumptions allow for very efficient constructions of advanced schemes like identity-based encryption [BF01], group signatures [CS97, BBS04], etc. none of these schemes yet have practical lattice-based realizations.

One of the major breakthroughs in lattice cryptography was the work of Gentry, Peikert, and Vaikuntanathan [GPV08], that showed how to use a short trap-door basis to generate short lattice vectors without revealing the trap-door by using an algorithm by Klein [Kle00]. In [GPV08], this was used to give the first lattice-based construction of secure hash-and-sign digital signatures and identity-based encryption schemes. This vector-sampling algorithm has since become a key component in many other lattice constructions, ranging from hierarchical identity-based encryption schemes [CHKP10, ABB10b] to standard-model signatures [ABB10b, Boy10b], attribute-based encryption [BGG+14] and many other constructions.

Unfortunately, even when using improved trap-doors [AP08, MP12] and instantiating with ideal lattices [LPR13a], signature schemes that used the GPV trapdoor approach were far less practical (by about two orders of magnitude) than the Fiat-Shamir ones [PDG14, DDLL13], and identity-based encryption had ciphertexts that were even longer - having ciphertexts on the order of millions of bits.[1]

[1]The only works that we are aware of that give actual parameters for candidate constructions that use trapdoor sampling are [MP12, RS10].

8.1.1 Our results

Our main result is showing that the GPV sampling procedure can in fact be used as a basis for practical lattice cryptography. The two main insights in our work are that one can instantiate the GPV algorithm using a particular distribution of NTRU lattices that have nearly-optimal trapdoor lengths, and that a particular parameter in the GPV algorithm can be relaxed, which results in shorter vectors being output with no loss in security. As our main applications, we propose identity-based encryption schemes that have ciphertext (and key) sizes of two and four kilobytes (for approximately 80-bit and 192-bit security, respectively) and digital signatures that have outputs (and keys) of approximately 5120 bits for about 192-bits of security. We believe that this firmly places GPV-based cryptographic schemes into the realm of practicality. The IBE outputs are orders of magnitude smaller than previous instantiations and the signature sizes are smaller by about a factor of 1.3 than in the previously shortest lattice-based scheme based on the same assumption [DDLL13].

Our schemes, like all other practical lattice-based ones, work over the polynomial ring $\mathbb{Z}_q[x]/(x^N + 1)$, where N is a power of 2 and q is a prime congruent to 1 mod $2N$. For such a choice of q, the polynomial $x^N + 1$ splits into N linear factors over $\mathbb{Z}_q$, which greatly increases the efficiency of multiplication over the ring. Our hardness assumption is related to the hardness, in the random oracle model, of solving lattice problems over NTRU lattices. These assumptions underlie the NTRU encryption scheme [HPS98], the NTRU-based fully-homomorphic encryption scheme [LTV12], and the recent signature scheme BLISS [DDLL13]. And even though this assumption is not related to the hardness of worst-case lattice problems via some worst-case to average-case reduction[2], in the fifteen years that the assumption has been around, there were no serious cryptanalytic threats against it. The work of [DDLL13] also provided experimental evidence that the computational complexity of finding short vectors in these special NTRU lattices was consistent with the extensive experiments of Gama and Nguyen on more general classes of lattices [GN08], some of which are connected to the hardness of worst-case lattice problems.

We implemented our schemes in software (see Table 8.1), and most of the algorithms are very efficient. The slowest one is user key generation, but this procedure is not performed often. More important is the underlying encryption scheme, which in our case is the Ring-LWE scheme from [LPR13a, LPR13b], which already has rather fast hardware implementations [PG14b]. And as can be seen from the tables, decryption and encryption are very fast in software as well and compare very favorably to state-of-the-art implementations of pairing-based constructions.

The timings were performed on an Intel Core i5-3210M laptop with a 2.5GHz CPU and 6GB RAM. Our implementation relies on the NFLLib library, which allows fast implementation of lattice-based cryptography. More details on this library can be found on [ABFK14, GAGL15] but our implementation is unfortunately not publicly available at this time. A less efficient but open source implementation in C++ can be found on:

https://github.com/tprest/Lattice-IBE/

[2]The work of [SS11] showed a connection between problems on NTRU lattices and worst-case problems, but for choices of parameters that do not lead to practical instantiations.

Table 8.1: Comparing our IBE (GPV) with a recent implementation [Gui13] of the Boneh-Franklin scheme (BF).

Scheme	**GPV-80**	**GPV-192**	**BF-128**	**BF-192**
User Private key size	11 kbits	27 kbits	0.25 kbits	0.62 kbits
Ciphertext size	13 kbits	30 kbits	3 kbits	15 kbits
User Key Generation	**8.6 ms**	**32.7 ms**	**0.55 ms**	**3.44 ms**
Encryption	**0.016 ms**	**0.033 ms**	**7.51 ms**	**38.7 ms**
Decryption	**0.007 ms**	**0.012 ms**	**5.05 ms**	**32.7 ms**

Table 8.2: IBE scheme parameters (see Section 8.6).

Security parameter λ	**80**	**192**
Root Hermite factor [GN08] γ	1.0075	1.0044
Polynomial degree N	512	1024
Modulus q	$\approx 2^{23}$	$\approx 2^{27}$
User Public key size	13 Kbits	30 Kbits
User Private key size	11 Kbits	27 Kbits
Ciphertext size	13 Kbits	30 Kbits
Ciphertext expansion factor	26	30

8.1.2 Related Work

Following the seminal work of [GPV08], there were attempts to improve several aspects of the algorithm. There were improved trap-doors [AP08], more efficient trap-door sampling algorithms [Pei10, MP12], and an NTRU signature scheme proposal [SS11]. All these papers, however, only considered parameters that preserved a security proof to lattice problems that were known to have an average-case to worst-case connection. To the best of our knowledge, our work is the first that successfully utilizes GPV trapdoor sampling in practice.

8.1.3 Identity-Based Encryption Scheme

In a public-key IBE scheme, the public key of every user in the system is a combination of the master authority's public key along with an evaluation of a publicly-computable function on the user's name or i.d.. The private key of each user is then derived by the master authority by using his master private key. We now give a brief description of the IBE in this chapter, which is built by using the GPV algorithm to derive the user's private keys from an NTRU lattice [GPV08, SS11], and then using the Ring-LWE encryption scheme of [LPR13a, LPR13b] for the encryption scheme.

The master public key in the scheme will be a polynomial $\mathbf{h}$ and the private key will consist of a "nice basis" for the $2N$-dimensional lattice generated by the rows of $\mathbf{A}_{h,q} = \begin{pmatrix} -\mathcal{A}(\mathbf{h}) & \mathbf{I}_N \\ q\mathbf{I}_N & \mathbf{O}_N \end{pmatrix}$, where $\mathcal{A}(\mathbf{h})$ is the anti-circulant matrix whose i^{th} row consists of the coefficients of the polynomial $x^i * \mathbf{h} \bmod (x^N + 1)$. A user with identity id will have a public key consisting of $\mathbf{h}$ as well as $\mathbf{t} = H(id)$, where H is some publicly-known cryptographic hash function mapping into $\mathbb{Z}_q[x]/(x^N+1)$. The user's private key will consist

Table 8.3: Signature scheme parameters (see Section 8.6.7).

Security parameter λ	**80**	**192**
Root Hermite factor γ	1.0069	1.0042
Polynomial degree N	256	512
Modulus q	$\approx 2^{10}$	$\approx 2^{10}$
Signature size	2560 bits	5120 bits

of a small polynomial $\mathbf{s}_2$ such that $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = \mathbf{t}$, where $\mathbf{s}_1$ is another small polynomial (how one generates these keys is explicited in Alg. 8.2 in Section 8.3). Encryption and decryption will proceed as in the Ring-LWE scheme of [LPR13a]. To encrypt a message $\mathbf{m} \in \mathbb{Z}[x]/(x^N + 1)$ with binary coefficients, the sender chooses polynomials $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ with small coefficients and sends the ciphertext

$$(\mathbf{u} = \mathbf{r} * \mathbf{h} + \mathbf{e}_1, \mathbf{v} = \mathbf{r} * \mathbf{t} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathbf{m}).$$

To decrypt, the receiver computes $\mathbf{v} - \mathbf{u} * \mathbf{s}_2 = \mathbf{r} * \mathbf{s}_1 + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathbf{m} - \mathbf{s}_2 * \mathbf{e}_1$. If the parameters are properly set, then the polynomial $\mathbf{r} * \mathbf{s}_1 + \mathbf{e}_2 - \mathbf{s}_2 * \mathbf{e}_1$ will have small coefficients (with respect to q), and so the coordinates which will decrypt to 0 will be small, whereas the coordinates which will decrypt to 1 will be close to $q/2$. Notice that for decryption, it is crucial for the polynomial $\mathbf{r} * \mathbf{s}_1 + \mathbf{e}_2 - \mathbf{s}_2 * \mathbf{e}_1$ to have small coefficients, which requires $\mathbf{s}_1$ and $\mathbf{s}_2$ to be as small as possible.

While the above follows the usual encryption techniques based on LWE, we need a little tweak to make the security proof based on KL-divergence work (see Section 8.5), since this argument only applies to search problems (while CPA security is a decisional problem). To do so we use a key-encapsulation mechanism, that is we encrypt a random key $\mathbf{k}$ rather than $\mathbf{m}$, and then use it as a one-time-pad to send $\mathbf{m} \oplus H'(\mathbf{k})$ where H' is a hash function.

8.1.4 Interlude: A Hash-and-Sign Digital Signature Scheme

The first part of the above IBE is actually a hash-and-sign digital signature scheme, as illustrated by Figure 8.1. The public (verification) key corresponds to the master authority's public key, the private (signing) key is the master private key, messages correspond to user i.d.'s, and signatures are the user private keys. To sign a message $\mathbf{m}$, the signer uses his private key to compute short polynomials $\mathbf{s}_1, \mathbf{s}_2$ such that $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = H(\mathbf{m})$, and transmits $\mathbf{s}_2$. The verifier simply checks that $\mathbf{s}_2$ and $H(\mathbf{m}) - \mathbf{h} * \mathbf{s}_2$ are small polynomials.

In the IBE, the modulus q is set deliberately large to avoid decryption errors, but this is not an issue in the signature scheme. By selecting a much smaller q, which allows one to sample from a tighter distribution, the signature size can be made more compact than the user private key size in the IBE.

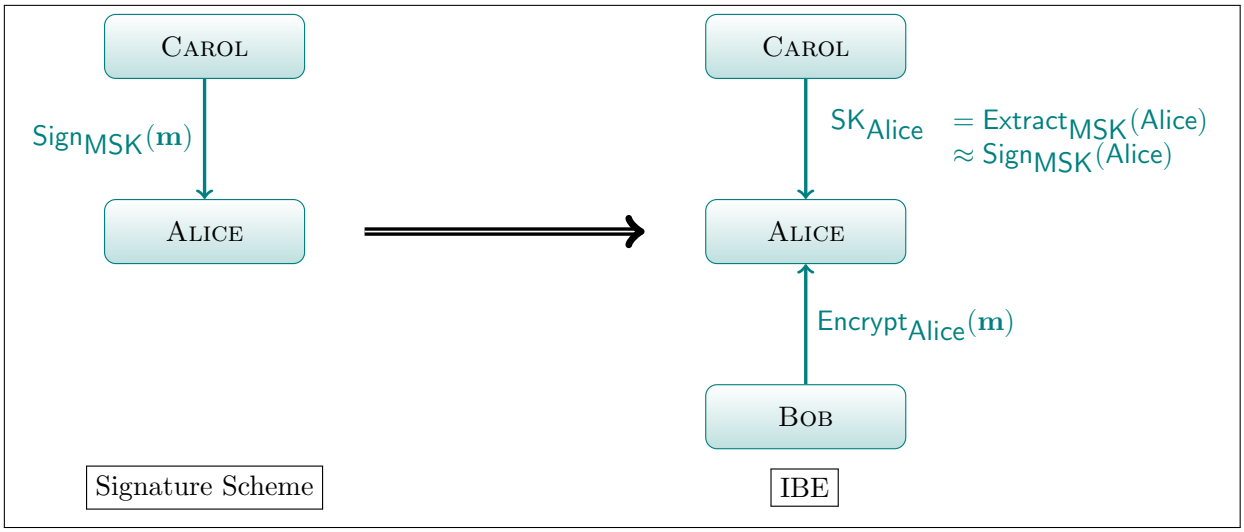


Figure 8.1: From a signature scheme to an IBE scheme

We can reprise our analysis from Chapter 6, the only thing that we add is concrete parameters for such a signature scheme in Table 8.3. The size of the keys and signatures compare very favorably to those of the BLISS signature scheme [DDLL13]. For example, for the 192 bit security level, the signature size in BLISS is approximately 6500 bits, whereas signatures in this work are approximately 5000 bits. In fact, further improvements to the signature size may be possible *via* similar techniques that were used for BLISS.

The main drawback of the hash-and-sign signature scheme is that signing requires sampling a discrete Gaussian over a lattice, whereas the Fiat-Shamir based BLISS scheme only required Gaussian sampling over the integers. At this point, the signature scheme in this chapter yields smaller signatures but BLISS is much faster. Since both BLISS and this current proposal are very new, we believe that there are still a lot of improvements left in both constructions.

8.1.5 Techniques and Chapter Organization

The main obstacle in making the above schemes practical is outputting short $\mathbf{s}_1, \mathbf{s}_2$ such that $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = \mathbf{t}$ while hiding the trap-door that allows for this generation. [GPV08] provided an algorithm where the norms of $\mathbf{s}_1, \mathbf{s}_2$ crucially depend on the length of the Gram-Schmidt orthogonalized vectors in the trap-door basis of the public lattice. In the previous Chapter 6 we showed, by experimental evidence backed up by a heuristic argument, that there exist distributions of NTRU lattices that have trap-doors whose lengths are within a small factor of optimal. Once we have such short trap-doors (which correspond to the master private key in the IBE), we can use the GPV algorithm to sample $\mathbf{s}_1, \mathbf{s}_2$ such that $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = \mathbf{t}$. In order for $(\mathbf{s}_1, \mathbf{s}_2)$ to reveal nothing about the trap-door, it's important that $\mathbf{s}_1, \mathbf{s}_2$ come from a distribution such that seeing $(\mathbf{h}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t})$ does not reveal whether $\mathbf{s}_1, \mathbf{s}_2$ were generated first and then $\mathbf{t}$ was computed as $\mathbf{s}_1 + \mathbf{h} * \mathbf{s}_2 = \mathbf{t}$, or whether $\mathbf{t}$ was first chosen at random and then $\mathbf{s}_1, \mathbf{s}_2$ were computed using the GPV sampler.

To prove this, [GPV08] showed that the distribution of $\mathbf{s}_1, \mathbf{s}_2$ produced by their sampler is statistically-close to some trapdoor-independent distribution. In Chapter 3, we showed that the requirement of statistical closeness can be relaxed, and we can instead use Kullback-Leibler divergence to obtain private keys shorter by a factor $\sqrt{2}$.

8.2 Preliminaries

As this is the last technical chapter, most of the tools have already been previously defined. We give a quick reminder for the most important of them.

8.2.1 Identity-Based Encryption

In the model of Boneh and Franklin [BF01], an IBE scheme is composed of one authority having extended powers, at least two users and four algorithms:

- Setup: This algorithm is run only one by the authority who creates the IBE environment. It creates the master private key MSK, which is kept secret, as well as a master public key MPK, which is rendered public to all users of the IBE scheme and may consist of a set of parameters including the message space and ciphertext space.

- $\mathsf{Extract}(\mathsf{MSK}, \mathsf{id})$: Given an identity $\mathsf{id} \in \{0,1\}^\star$, the authority (an no one else) uses its master private key to derive a private key $\mathsf{SK}_{\mathsf{id}}$ associated to id.

- $\mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, \mathsf{m})$: To send a message m to id, anyone can encrypt it using only the master public key MPK and the identity id.

- $\mathsf{Decrypt}(\mathsf{SK}_{\mathsf{id}}, \mathsf{C})$: Given a private key $\mathsf{SK}_{\mathsf{id}}$ associated to an identity id, anyone (presumably the owner of the identity, or in some cases the authority) can decrypt a message C sent to id.

8.2.2 Notations

For the rest of the chapter, N will be a power-of-two integer. We will work in the cyclotomic ring $\mathcal{R} \triangleq \mathbb{Z}[x]/(\phi_{2N}(x)) = \mathbb{Z}[x]/(x^N+1)$ (and occasionally the field $\mathbb{K} \triangleq \mathbb{Q}[x]/(x^N+1)$).

We recall that most of the notation conventions are stated in pages 11 to 19. Let $f = \sum_{i\in\mathbb{Z}_N} f_i x^i$ and $g = \sum_{i\in\mathbb{Z}_N} g_i x^i$ in $\mathbb{K}$.

- $r(f)(x) = (x \cdot f(x)) \bmod \phi_{2N}$. r is called a rotation because it results in an almost perfect circular rotation of the coefficients of f : $r(f)(x) = -f_{N-1} + \sum_{i=1}^{N-1} f_{i-1}x^i$. As a side note, r is an isometry and thus allows to use the techniques discussed in Chapter 4.

- (f,g) is the vector whose coefficients are $f_0, ..., f_{N-1}, g_0, ..., g_{N-1}$.

- $f^\star(x) = f(1/x) \bmod (x^N+1) = f_0 - \sum_{i\in\mathbb{Z}_N} f_i x^{(-i \bmod N)}$ is the conjugate of f in $\mathbb{K}$.

- $\lfloor f \rceil$ is the coefficient-wise rounding of f. The same notation applies for vectors.

As many other cryptographic constructions over ring lattices, our IBE scheme gains efficiency from the duality vector-polynomial. To avoid any ambiguity, when considering two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{K} \cong \mathbb{Q}^N$, their dot product as vectors is written $\langle \mathbf{u}, \mathbf{v} \rangle$, consistently with the notations of the previous chapters, and their multiplication as elements of $\mathbb{K}$ – that is, as polynomials $\bmod (x^N+1)$ – is written $\mathbf{u} * \mathbf{v}$.

8.2.3 Anticirculant matrices

We recall that for N a power of two and $f \in \mathbb{R}[x]$ $\mathcal{A}_{\phi_{2N}}(f)$ is the N-dimensional anticirculant matrix defined by:

$$\mathcal{A}_{\phi_{2N}}(f) = \begin{pmatrix} f_0 & f_1 & f_2 & \ddots & f_{N-1} \\ -f_{N-1} & f_0 & f_1 & \ddots & f_{N-2} \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ -f_1 & -f_2 & \ddots & \ddots & f_0 \end{pmatrix} = \begin{pmatrix} (f) \\ (x * f) \\ \vdots \\ (x^{N-1} * f) \end{pmatrix}$$

When N is clear from context, we just write $\mathcal{A}(f)$. Anticirculant matrices verify the following properties:

1. $\mathcal{A}(f) + \mathcal{A}(g) = \mathcal{A}(f + g)$, and $\mathcal{A}(f) \times \mathcal{A}(g) = \mathcal{A}(f * g)$
2. $\mathcal{A}_{\phi_{2N}}(f)^t = \mathcal{A}_{\phi_{2N}}(f^\star)$

Complementary information can be found in Section 4.2.4.

8.2.4 Gaussian Sampling

At this point, we assume that readers are familiar with Gaussian sampling and the various algorithms developed to perform it. We recall that it was introduced in [GPV08] as a technique to use a short basis as a trap-door without leaking any information about the short basis; in particular it provably prevents any attack in the lines of [NR06, DN12b] designed against the NTRUSign scheme. In this chapter, we will use Gaussian sampling to sample user keys for our IBE scheme, in the same way we used it to sample signatures in Chapter 5, Section 6.2.

8.2.5 Hardness Assumption

We can base the hardness of our IBE scheme on two assumptions that have been previously used in the literature. The first assumption deals with NTRU lattices and states that if we take two random small polynomials $f, g \in \mathcal{R}_q$, their quotient $\mathbf{h} = g/f$ is indistinguishable from a sample from the uniform distribution in $\mathcal{R}_q$. This assumption was first formally stated in [LTV12], but it has been studied since the introduction of the NTRU cryptosystem [HPS98] in its computational form (i.e. recovering the polynomials f and g from $\mathbf{h}$). Despite more than fifteen years of cryptanalytic effort, there has not been any significant algorithmic progress towards solving either the search or decision version of this problem. As a side note, Stehlé and Steinfeld [SS11] showed that for large enough f and g generated from a discrete Gaussian distribution, the quotient g/f is actually uniform in $\mathcal{R}_q$. Thus if one were to use larger polynomials, the NTRU assumption would be unnecessary. Using smaller polynomials, however, results in much more efficient schemes.

The second assumption we will be using is the Ring-LWE assumption [LPR13a] stating that the distribution of $(\mathbf{h}_i, \mathbf{h}_i * \mathbf{s} + \mathbf{e}_i)$, where $\mathbf{h}_i$ is random in $\mathcal{R}_q$ and $\mathbf{s}, \mathbf{e}_i$ are small polynomials, is indistinguishable from uniform. When the number of such samples given is polynomial (with respect to the degree of $\mathbf{s}$), the coefficients of $\mathbf{e}_i$ cannot be too small [AG11], however, if we only give one or two samples (as is done for Ring-LWE encryption), there have been no specific attacks found if the coefficients of $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2$ are taken from a

very small set like $\{-1, 0, 1\}$. In our work, we choose to sample them from such a small set, but the scheme can be changed to sample from any other slightly larger set at the expense of slightly increasing the size of the modulus.

8.3 The IBE Scheme

In this section, we present our IBE scheme.

Algorithm 8.1 $\mathsf{Setup}(N, q)$

Require: N, q
Ensure: Master Private Key $\mathbf{B} \in \mathbb{Z}_q^{2N \times 2N}$ and Master Public Key $\mathbf{h} \in \mathcal{R}_q$

1: $\sigma_f = 1.17\sqrt{\frac{q}{2N}}$ \\ σ_f chosen s.t. $\mathbb{E}[\|\mathbf{b}_1\|] = 1.17\sqrt{q}$
2: $f, g \leftarrow D_{\mathbb{Z}_N, \sigma_f}$
3: Norm $\leftarrow \max\left(\|(g, -f)\|, \left\|\left(\frac{qf^\star}{f*f^\star+g*g^\star}, \frac{qg^\star}{f*f^\star+g*g^\star}\right)\right\|\right)$ \\ We compute $|\tilde{\mathbf{B}}_{f,g}|$
4: if Norm$> 1.17\sqrt{q}$, go to step 2
5: Using the extended Euclidean algorithm, compute $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ s.t.
   - $\rho_f * f = R_f$
   - $\rho_g * g = R_g$
6: if $GCD(R_f, R_g) \neq 1$ or $GCD(R_f, q) \neq 1$, go to step 2
7: Using the extended Euclidean algorithm, compute $u, v \in \mathbb{Z}$ s.t. $u \cdot R_f + v \cdot R_g = 1$
8: $F \leftarrow qv\rho_g, G \leftarrow -qu\rho_f$
9: $k = \left\lfloor \frac{F*f^\star+G*g^\star}{f*f^\star+g*g^\star} \right\rceil \in \mathcal{R}$
10: Reduce F and G:
   - $F \leftarrow F - k * f$
   - $G \leftarrow G - k * g$
11: $\mathbf{h} = g * f^{-1} \bmod q$
12: $\mathbf{B} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$

Algorithm 8.1 generates a short basis $\mathbf{B}$ of $\Lambda_{\mathbf{h},q}$, making it a trapdoor for sampling short elements $(\mathbf{s}_1, \mathbf{s}_2)$ such that $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = \mathbf{t}$ for any $\mathbf{t}$, without leaking any information about itself. Steps 5 to 12 of the algorithm is just completing (f, g) into a full NTRU basis $\mathbf{B}$.

Algorithm 8.2 $\mathsf{Extract}(\mathbf{B}, \mathsf{id})$

Require: Master secret key $\mathbf{B} \in \mathbb{Z}_q^{2N \times 2N}$, standard deviation $\sigma \geqslant |\tilde{\mathbf{B}}| \cdot \eta'_\epsilon(\mathbb{Z})$, hash function $H : \{0,1\}^\star \to \mathbb{Z}_q^N$, user identity id
Ensure: User secret key $\mathsf{SK}_{\mathsf{id}} \in \mathcal{R}_q$

1: **if** $\mathsf{SK}_{\mathsf{id}}$ is in local storage **then**
2: Output $\mathsf{SK}_{\mathsf{id}}$ to user id
3: **else**
4: $\mathbf{t} \leftarrow H(\mathsf{id}) \in \mathbb{Z}_q^N$
5: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow (\mathbf{t}, \mathbf{0}) - \mathsf{KleinSampler}(\mathbf{B}, \sigma, (\mathbf{t}, \mathbf{0}))$ \\ $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = \mathbf{t}$
6: $\mathsf{SK}_{\mathsf{id}} \leftarrow \mathbf{s}_2$
7: Output $\mathsf{SK}_{\mathsf{id}}$ to user id and keep it in local storage
8: **end if**

Here the Extract is exactly the Sign procedure of Section 6.3, Algorithm 6.2, with Klein's sampler as the chosen sampling procedure.

Algorithm 8.3 Encrypt$(\mathsf{id}, \mathbf{m})$

Require: Hash functions $H : \{0,1\}^\star \to \mathbb{Z}_q^N$ and $H' : \{0,1\}^N \to \{0,1\}^N$, message $\mathbf{m} \in \{0,1\}^N$, master public key $\mathbf{h} \in \mathcal{R}_q$, identity id
Ensure: Encryption $(\mathbf{u}, \mathbf{v}, \mathbf{c}) \in \mathcal{R}_q^2$ of $\mathbf{m}$ under the public key of id
1: $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \{-1, 0, 1\}^N$; $\mathbf{k} \leftarrow \{0,1\}^N$ (uniform)
2: $\mathbf{t} \leftarrow H(\mathsf{id})$
3: $\mathbf{u} \leftarrow \mathbf{r} * \mathbf{h} + \mathbf{e}_1 \in \mathcal{R}_q$
4: $\mathbf{v} \leftarrow \mathbf{r} * \mathbf{t} + \mathbf{e}_2 + \lfloor q/2 \rfloor \cdot \mathbf{k} \in \mathcal{R}_q$
5: Drop the least significant bits of $\mathbf{v}$: $\mathbf{v} \leftarrow 2^\ell \left\lfloor v/2^\ell \right\rceil$
6: Output $(\mathbf{u}, \mathbf{v}, \mathbf{m} \oplus H'(\mathbf{k}))$

Note that encryption is designed using a key-encapsulation mechanism; the hash of the key $\mathbf{k}$ is used to one-time-pad the message. If H' is modeled as a random oracle, this makes the CPA security (a decisional problem) of the scheme as hard as finding the key $\mathbf{k}$ exactly (a search problem). Basing the security argument on a search problem is necessary for our KL Divergence-based security argument to hold, as explained in Section 3.2.1.

Before sending the ciphertext $(\mathbf{u}, \mathbf{v}, \mathbf{m} \oplus H'(\mathbf{k}))$, we drop the the least significant bits of $\mathbf{v}$ in step 5 to save space. However, it won't have any practical impact over the scheme.

Algorithm 8.4 Decrypt$(\mathsf{SK}_{\mathsf{id}}, (\mathbf{u}, \mathbf{v}, \mathbf{c}))$

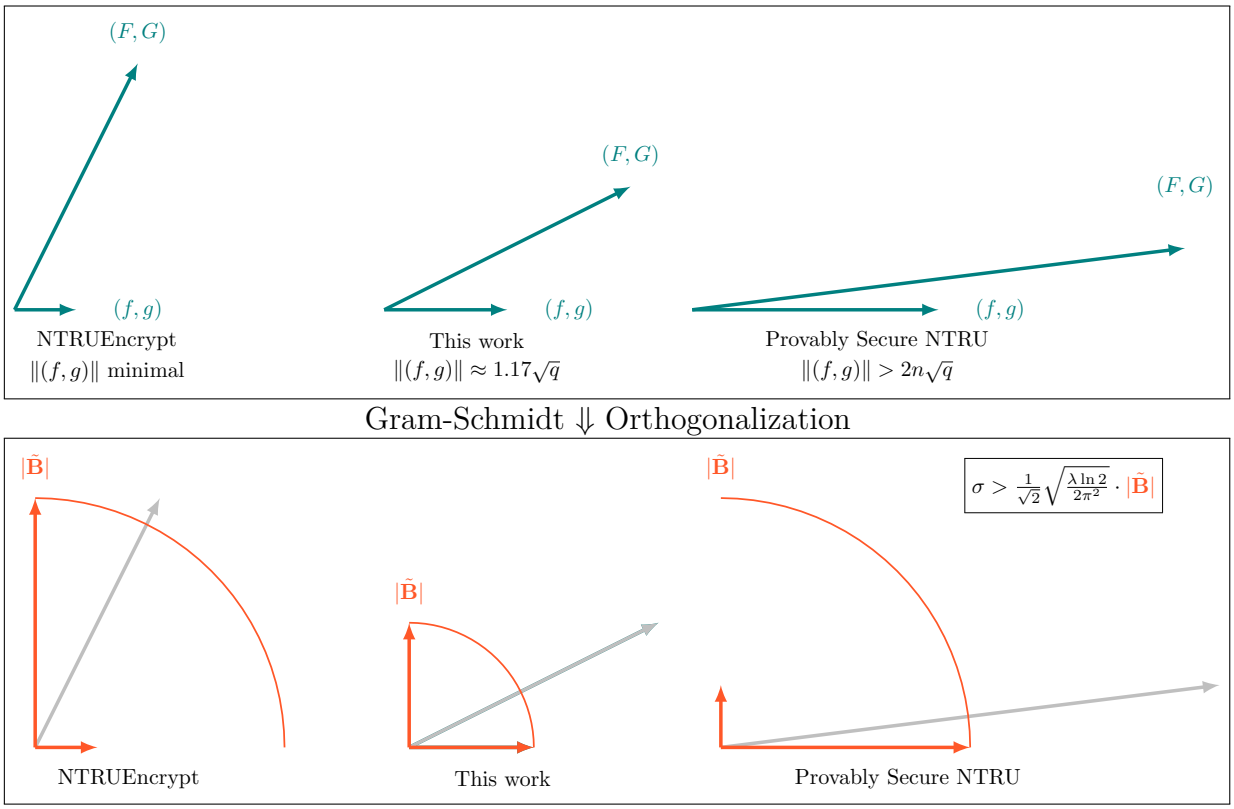
Require: User secret key $\mathsf{SK}_{\mathsf{id}}$, encryption $(\mathbf{u}, \mathbf{v}, \mathbf{c})$ of $\mathbf{m}$
Ensure: Message $\mathbf{m} \in \{0,1\}^N$
1: $\mathbf{w} \leftarrow \mathbf{v} - \mathbf{u} * \mathbf{s}_2$
2: $\mathbf{k} \leftarrow \left\lfloor \frac{\mathbf{w}}{q/2} \right\rceil$
3: Output $\mathbf{m} \leftarrow \mathbf{c} \oplus H'(\mathbf{k})$

In Algorithm 8.4, if the message $\mathbf{m}$ to be decrypted is much shorter than N bits, then we can use the remaining bits to embed $\mathbf{m}$ into an error-correcting code. This allows a small number of decryption errors, which in turn allows to reduce the modulus q by a few bits. This is discussed in Section 8.5.3.

8.4 Optimizing the Setup and the Extract

In this section, we first optimize the Setup and Extract procedures of our IBE. We recall that these are exactly the KeyGen and Sign procedures of the Full-Domain Hash (FDH) signature scheme presented in Section 6.3.

The IBE is affected in the same way as the FDH signature scheme by the performance of these algorithms: the shorter the key (resp. signature in the FDH scheme) sampled, the more secure the scheme. Therefore we ignore for now the Encrypt/Decrypt parts. To optimize space and security, we have two leverages: the choice of the family of lattices and of the sampler.



Our choice of distribution for NTRU lattices is very different from preexisting ones. We note that there is no distribution "better" than the others: the one from the NTRUEncrypt scheme [HPS98] is used for different purposes, ours optimizes Gaussian sampling and the one from Stehlé and Steinfeld [SS11] aims at provable security.

Figure 8.2: Comparison with other choices of distributions for NTRU lattices

- For the family of lattices, we choose the NTRU lattices for the reasons previously given: they have a very compact representation and have resisted extended cryptanalysis over the last years.

- For the choice of the sampler, we can neglect its running time since the Extract is realized only once per user, as opposed to an arbitrary number of Encrypt/Decrypt. Without this constraint, Klein's sampler (or its fast Fourier variant described in Chapter 7) becomes the only logical choice when opposed to the Hybrid and Peikert's.

Fortunately, we already analyzed the use of Klein's sampler with NTRU lattices in Section 6.2. It showed that taking $\|(f, g)\| \approx 1.17\sqrt{q}$ allows to sample very short vectors using Klein's sampler with NTRU lattices: the standard deviation σ of the vectors we sampled are within a factor 1.17 of the best theoretical σ known for Klein's sampler, and both our heuristic and experiments suggest that it is the best achievable in practice for NTRU lattices.

We also have to compute $|\tilde{\mathbf{B}}_{f,g}|$ in practice. For general lattices, this is done by applying the Gram-Schmidt process to the basis and computing the maximum length of the resulting vectors. However, in the case of NTRU lattices, Corollary 6.5 allows to compute $|\tilde{\mathbf{B}}_{f,g}|$ much faster, in time $O(N \log(N))$ instead of $O(N^3)$. Moreover, this verification step is done *before* completing (f, g) into a full NTRU basis (f, g, F, G). Since the latter operation

is by far the costliest part of the Setup, it can in practice be an appreciable feature to be able to check the "quality" of a private key (f, g, F, G) before completing it. We would also have this property if we used the hybrid sampler, but it doesn't seem to apply to Peikert's sampler.

8.4.1 Parameter Improvement in the Case of Bounded Queries

An idea brought up by Bai et al. in [BLL+15] is to consider the security of the signature scheme from [GPV08] for a finite number of queries. Combined with replacing the statistical distance by the Rényi divergence, this allows them to use the GPV signature scheme with a much smaller standard deviation.

Their analysis transfers seamlessly to the extract part of our IBE scheme, and if there is one situation where it is perfectly reasonable to consider as bounded the number of queries, it is precisely IBE. For most of the applications of IBE – like secure mail, secure phones, wireless sensors and botnets which are all detailed in the next Chapter 9 – it is reasonable to assume that no more than 2^{20} user key extracts are performed, and 2^{33} (more than the current number of living humans on Earth) seems almost out of reach.

Therefore, taking the number of extract queries as a variable can be used as a leverage to further reduce the parameters of the IBE. Expanding on the idea of [BLL+15] might even lead to a reduction of the required precision. However, we don't elaborate on both ideas and leave this for future work.

8.5 Optimizing the Encryption and Decryption

In this section, we explain a few choices in our scheme related to the encryption and decryption parts. Some choices are related to security, some others for optimization, and one serves both.

First, Section 8.5.1 evaluate how big the modulus q must be set to ensure that users decrypt correctly. Then, in Section 8.5.2, we explain the security reason behind the use of a key-encapsulation mechanism in the encryption. We elaborate how to reduce the ciphertexts' size using bit dropping in Section 8.5.3. To finish, Section 8.5.4 considers using error correcting codes to gain about an extra 15% on the size of the ciphertexts.

In addition, the scheme we described is only CPA-secure. Fortunately, works by Yang et al. [YKH+06] and Kitagawa et al. [KYH+06] have already documented how to make IND-CCA2 any IND-CPA IBE in the random-oracle model.

8.5.1 Correctness of Decryption

In order for an user to decrypt correctly, $\mathbf{y} = \mathbf{r} * \mathbf{s}_1 + \mathbf{e}_2 - \mathbf{e}_1 * \mathbf{s}_2$ must have all its coefficients in $(-\frac{q}{4}, \frac{q}{4})$[3], so we need to set q big enough. In practice, this gives $q \approx 2^{24}$ for $\lambda = 80$, and $q \approx 2^{27}$ for $\lambda = 192$.

Proof outline. We use Lyapunov's Central Limit Theorem to approximate each y_i with the Gaussian $\mathcal{N}(0, \sigma_y)$, where $\sigma_y = \sqrt{\frac{2}{3}(\|\mathbf{s}\|^2 + 1)} \approx 2\sigma\sqrt{\frac{N}{3}}$, since $\|\mathbf{s}\|$ is expected to be about $\sigma\sqrt{2N}$. Then we use a tailcut inequality for Gaussians.

[3] In fact we take $\frac{q}{8}$ instead of $\frac{q}{4}$ to account for bit dropping.

8.5.2 CPA Security and Kullback-Leibler Divergence

During this work, we used the KL divergence to argue that if a scheme is λ-bit secure with access to a perfect oracle $\mathcal{P}$, then it as also λ-bit secure with access to an imperfect oracle $\mathcal{Q}$ as long as $D_{\mathrm{KL}}(\mathcal{P}\|\mathcal{Q}) \leqslant 2^{-\lambda}$.

Compared to a statistical-based analysis, this allowed us to sample with a standard deviation smaller by a factor $\sqrt{2}$ and to cut the required precision in half.

However, the security argument we gave is tight only for search problems (see Remark 3.4). We can base the unforgeability of FDH schemes on it, since forging a signature is a search problem, but it is unclear if it could be directly applied to any CPA scheme: CPA security is a decisional problem, not a search problem.

Yet our IBE design makes this argument valid: we designed encryption using a key-encapsulation mechanism, the random key $\mathbf{k}$ being fed into a hash function H' to one-time-pad the message. Modeling H' as a random oracle, one easily proves that breaking CPA security with advantage p is as hard as recovering $\mathbf{k}$ with probability p, which is a search problem.

8.5.3 Drop the Bit

The step 5 of Algorithm 8.3 drops the least ℓ significant bits of $\mathbf{v}$. The goal of this operation is not to make the scheme more secure (although this is arguably the case since it gives an attacker less information), but simply to reduce the size of the ciphertexts.

Of course dropping bits can also lead to incorrect decryption. However, for $\ell \leqslant \lfloor \log_2 q \rfloor - 3$, it doesn't significantly affect the correct decryption rate of the scheme, so we take this value of ℓ as standard.

This allows to divide the size of the ciphertexts by a bit less than 2.

8.5.4 Error-Correcting Codes

To further reduce the size of the ciphertexts, we can resort to error-correcting codes (or ECC). This idea was first proposed in [GFS+12], but to the best of our knowledge no one developed it – including the authors of the aforementioned paper.

Normally, the modulus q needs to be set big enough so that all the bits of $\mathbf{y}$ are smaller (in absolute value) than $\frac{q}{8}$. However, in practice public-key cryptography is typically used to encapsulate a private key or to proceed to a key-exchange, and IBE is no exception to this rule (see Chapter 9). In these conditions, it is reasonable to assume that only the first λ bits of the message store relevant information, therefore only these need to be correctly decrypted.

In addition, the $N - \lambda$ remaining bits can be used to store redundant information about the encapsulated key. In other words, one can use a correcting code $C : \mathbb{F}_2^\lambda \mapsto \mathbb{F}_2^N$ to encode the information $\mathbf{i}$ being sent.

Let $C : \mathbb{F}_2^\lambda \mapsto \mathbb{F}_2^n$ a $[n, k, d]$-error correcting code, that is a correcting-code encoding k bits of information into a n-bits codeword, whose Hamming distance is d. C is able to detect at most $d - 1$ errors and to correct at most $\lfloor d/2 \rfloor$ errors. We consider three scenarios:

- **Plain.** The information $\mathbf{i} \in \mathbb{F}_2^\lambda$ is simply encoded on the first λ bits of $\mathbf{m}$. We require that these λ first bits all decrypt correctly with overwhelming probability.

Table 8.4: Comparing how small $\lfloor \log_2 q \rfloor$ can be set, in function of the parameters of the scheme and the encoding of the messages.

Encoding / Parameters	Plain	Constrained ECC			Relaxed ECC		
		$d = 7$	$d = 17$	$d = 45$	$d = 7$	$d = 17$	$d = 45$
$\lambda = 192, N = 1024$	27	25	24	23	24	23	22
$\lambda = 80, N = 512$	24	22	21	20	21	20	19

- **Constrained ECC.** The information $\mathbf{i}$ is encoded into a codeword $C(\mathbf{i}) \in \mathbb{F}_2^n$ using an error-correcting code C that can correct $\lfloor d/2 \rfloor$ errors. We require that at most $\lfloor d/2 \rfloor$ errors occur with overwhelming probability. In this setting, we don't lose in generality compared to the plain setting.
- **Relaxed ECC.** This setting is the same as constrained ECC, except that we require the numbers of errors to be at most $\lfloor d/2 \rfloor$ with high probability (say more than $1-2^{-10}$), instead of overwhelming. However, we require to be negligible the probability that more than $d-1$ errors occur. Meeting both these conditions allows Alice to ask Bob to resend her the information $\mathbf{i}$ in the (rare) case that she cannot decrypt it correctly, while making sure (with overwhelming probability) that she cannot decrypt incorrectly without her knowledge.

These three scenarios allow for a various number of decryption errors, which leads to various sizes for the modulus q. This is summarized in Table 8.4.

8.6 Security analysis of the IBE scheme

We now proceed to analyze the security of the scheme. If we only consider the setup and the extraction, then the security of the scheme is equivalent to that of the underlying signature scheme. This scheme was already studied in Chapter 6. In particular, equations 6.3 and 6.4 tell us which parameters we can choose so that the scheme isn't vulnerable to user's key recovery through lattice attacks.

However, the most vulnerable part of our IBE scheme will be the actual encryption. Still, we will first run through the best attacks on the master public key and user private keys because these correspond exactly to attacks on the key and signature forgery, respectively, in the hash-and-sign digital signature scheme.

Our master public key polynomial $\mathbf{h}$ is not generated uniformly at random, but rather as $g * f^{-1}$. The best-known attack for distinguishing an NTRU polynomial from a random one is to find the polynomials f, g that are "abnormally short". This involves finding the short f and g such that $\mathbf{h} * f - g = 0 \bmod q$. This is equivalent to finding the vector (f, g) in a $2N$-dimensional lattice with determinant q^N. From Sections 6.4.1 and 6.5.1, we know that the euclidean norm of the vector (f, g) is approximately $1.17\sqrt{q}$ and so calculating the value of γ using (6.4), we get

$$\frac{\sqrt{2N/(2\pi e)} \cdot \sqrt{q}}{1.17\sqrt{q}} = .4\gamma^{2N} \Longrightarrow \gamma = (\sqrt{N}/1.368)^{1/2N},$$

which is 1.0054 for $N = 256$ and 1.0027 for $N = 512$, which is already beyond the realm of practical algorithms. The private user keys $(\mathbf{s}_1, \mathbf{s}_2)$ are generated with standard deviation

of about $\sigma = 1.17\eta'_\epsilon(\mathbb{Z}) \cdot \|\tilde{\mathbf{B}}\|$, which gives $\sigma \approx 1.5\sqrt{q}$ for $N = 256$ (resp. $\sigma \approx 2.24\sqrt{q}$ for $N = 512$), and so the vector has length $\sigma\sqrt{2N}$, which by (6.3) results in a value of γ,

$$\frac{\sigma\sqrt{2N}}{\sqrt{q}} = \gamma^{2N} \Longrightarrow \begin{cases} \gamma = (2.137\sqrt{N})^{1/2N} & \text{for} \quad N = 256 \\ \gamma = (3.162\sqrt{N})^{1/2N} & \text{for} \quad N = 512 \end{cases}$$

which is 1.0069 for $N = 256$ and 1.0042 for $N = 512$.

We now move on to the hardness of breaking the CPA-security of the scheme. Encryption (disregarding the message) consists of $(\mathbf{u} = \mathbf{r} * \mathbf{h} + \mathbf{e}_1, \mathbf{v} = \mathbf{r} * \mathbf{t} + \mathbf{e}_2)$, where the coefficients of $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ have coefficients chosen from $\{-1, 0, 1\}$. In order to avoid decryption errors, the value of the modulus q has to be set fairly high (see Table 8.1). The best-known attack against the encryption scheme involves essentially recovering the errors $\mathbf{e}_1, \mathbf{e}_2$. From the ciphertext $(\mathbf{u}, \mathbf{v})$, we can set up the equation $(\mathbf{t} * \mathbf{h}^{-1}) * \mathbf{e}_1 - \mathbf{e}_2 = (\mathbf{t} * \mathbf{h}^{-1}) * \mathbf{u} - \mathbf{v} \bmod q$, which can be converted into the problem of finding the $2N+1$-dimensional vector $(\mathbf{e}_1, \mathbf{e}_2, 1)$ in a $2N + 1$-dimensional lattice with determinant q^N. Using (6.4), we get

$$\frac{\sqrt{2N/(2\pi e)} \cdot \sqrt{q}}{\|(\mathbf{e}_1, \mathbf{e}_2, 1)\|} = .4\gamma^{2N} \Longrightarrow \gamma = (.74\sqrt{q})^{1/2N},$$

which gives us $\gamma = 1.0075$ for $N = 512$ and $\gamma = 1.0044$ for $N = 1024$.

The attacks we just described are the best known against our scheme. For completeness, we list a few other attacks in the rest of this section.

8.6.1 Master Secret Key recovery by Brute-force and Meet-in-the-Middle

Brute-force key recovery. This is the most basic and full-frontal attack : the attacker samples a polynomial $f^\star$ of norm $\leqslant \sqrt{0.6q}$, computes $g^\star = \mathbf{h} * f^\star$ and hopes that $\|(f^\star, g^\star)\| \leqslant 1.17\sqrt{q}$. He will then have obtained with very high probability (f, g) or one of its rotation $r^j(f, g)$.

f is sampled from $\mathcal{D}^N_{\mathbb{Z},\sigma_f}$, with $\sigma_f = \sqrt{\frac{3q}{5N}}$. For each i, $\mathbb{P}(f_i = f_i^\star) \leqslant \frac{1}{\sigma_f\sqrt{2\pi}} = \sqrt{\frac{6\pi q}{5N}}$, so the randomly sampled polynomial $f^\star$ will actually be equal to f with probability at most $\left(\frac{5N}{6\pi q}\right)^{N/2}$. Doing the same analysis for each of the $r^j(f)$ ensures us that the attack will succeed with probability at most $N\left(\frac{5N}{6\pi q}\right)^{N/2}$.

Meet-in-the-Middle key recovery. Odlyzko described a Meet-in-the-Middle attack against NTRUEncrypt, which also applies to [DDLL13] and the present scheme. The details can be found in [HG07], and the attack runs in time equal to the square root of the brute-force attack running time. The cost of this attack in time and memory is equal to about

$$\frac{1}{\sqrt{N}}\left(\frac{6\pi q}{5N}\right)^{N/4}$$

8.6.2 Master Secret Key recovery by primal lattice attack

This attack consists in applying lattice reduction to $\Lambda_{h,q}$ and hope to find the private key as a short vector. As already explained in [DDLL13], experiments show that even

though there are N short vectors $(f,g), r(f,g), ..., r^{N-1}(f,g)$, it doesn't affect the behavior of BKZ, and therefore also use the BKZ 2.0 methodology. This ensures us that the scheme is secure against primal lattice attack provided that

$$\sqrt{\frac{q \cdot 2N}{2\pi e}} < 0.4\gamma^{2N}\|(f,g)\|$$

Since $\|(f,g)\| = 1.17\sqrt{q}$, this gives us $\gamma > \left(\frac{N}{0.192\pi e}\right)^{1/4N}$

8.6.3 User private key forgery with approached collision attack

This attack is very similar to the one presented in [HGHPW05]. In this attack, $\mathcal{A}$ chooses a number of $\mathbf{s}_2^i \leftarrow \mathcal{D}_1$, computes $\mathbf{s}_1^i = H(id) - \mathbf{h} * \mathbf{s}_2^i$ and hopes that one of the $\mathbf{s}^i \triangleq \begin{pmatrix}\mathbf{s}_1^i \\ \mathbf{s}_2^i\end{pmatrix}$ has its norm less than $\|\mathbf{s}\|_{max} = 2\sigma\sqrt{\pi N}$, where $\sigma = \eta'_\epsilon(\mathbb{Z}) \cdot \sqrt{1.2q}$. Assuming the attacker cannot control the norm of $\mathbf{s}_1^i$ more than if he was sampling a random vector in $\mathbb{Z}_q^N$, the probability that

$$\mathbb{P}\left(\|x\| \leqslant \|\mathbf{s}\|_{max} | x \xleftarrow{\$} \mathbb{Z}_q^N\right) \approx \frac{Vol(\mathcal{B}_N(2\sigma\sqrt{\pi N}))}{Vol(\mathbb{R}_q^N)} \approx \frac{1}{\sqrt{\pi N}} \cdot \left(\frac{4\pi^2\sigma^2 e}{q^2}\right)^{N/2}$$

The attacker can also try to find "collisions" that yields valid user private keys : if $(\mathbf{s}_1^i, \mathbf{s}_2^i)$ and $(\mathbf{s}_1^j, \mathbf{s}_2^j)$ verify the equation

$$\mathbf{s}_1 = H(id) - \mathbf{h} * \mathbf{s}_2 \qquad (1)$$

then $(\mathbf{s}_1^i + \mathbf{s}_1^j - H(id), \mathbf{s}_2^i + \mathbf{s}_2^j)$ also verifies (1). It then suffices that $\mathbf{s}_1^i$ and $H(id) - \mathbf{s}_1^j$ are at a close distance for $(\mathbf{s}_1^i + \mathbf{s}_1^j - H(id), \mathbf{s}_2^i + \mathbf{s}_2^j)$ to be a valid signature.

This attack is similar to the birthday attack, with the difference that the attacker don't look for exact collisions, but close neighbors, which is much more complicated to do : unlike in the exact collision case, the attacker cannot simply sort the $\mathbf{s}_1^i, H(id) - \mathbf{s}_1^i$ in order to find a collision.

However, we assume that there is an analogous algorithm for finding close neighbors, therefore assuming the attacker is stronger than he actually is. The attacker can then hope to find a collision with a number of samples equal to :

$$\mathbb{P}\left(\|x\| \leqslant \|\mathbf{s}\|_{max} | x \xleftarrow{\$} \mathbb{Z}_q^N\right)^{-1/2} \approx (\pi N)^{1/4} \cdot \left(\frac{q^2}{4\pi^2\sigma^2 e}\right)^{N/4}$$

Therefore, to get a security parameter λ, it is sufficient to set

$$q \geqslant 2^{\frac{4\lambda}{N}+7}(\eta'_\epsilon)^2(\mathbb{Z})$$

8.6.4 User private key recovery by solving SIS

As our security proof explains, being able to forge an user private key implies being able to solve the NTRU-SIS problem.

If one is able to attain a Hermite factor γ, then it is hard to forge user private keys provided that the maximum accepted norm $\|\mathbf{s}\|_{\max}$ for user private keys verify

$$\|\mathbf{s}\|_{\max} \leqslant \gamma^{2N}\sqrt{q}$$

In practice, $\|\mathbf{s}\|_{\max} = \eta'_\epsilon(\mathbb{Z})\sqrt{4.8q\pi N}$, so this gives us $\gamma \geqslant \left(\eta'_\epsilon(\mathbb{Z})\sqrt{4.8\pi N}\right)^{1/2N}$. We note that this is a stronger condition than the one from Section 8.6.2.

8.6.5 Plaintext message recovery via Brute-force and Meet-in-the-Middle

Brute-force message recovery. This attack is almost exactly like the one from Section 8.6.1. Having intercepted an encrypted message $(\mathbf{u}, \mathbf{v})$, the attacker samples a polynomial $\mathbf{r}^\star$ and hopes that $\|\mathbf{h} * \mathbf{r} - \mathbf{u}\|$ is small.

Let each coefficient of $\mathbf{r}, \mathbf{e}_1$ be in $\{-1, 0, 1\}$, and equal to 0 with probability p, and to 1 (resp. -1) with probability $\frac{1}{2}(1-p)$. Then $\sigma_r = \sqrt{\frac{2}{3}(1-p)}$

Re-using the same analysis, this attack succeeds with probability at most Np^N.

Meet-in-the-Middle message recovery. The MitM attack applies once again, and its cost in time and memory is equal to about $\frac{1}{\sqrt{N}} p^{-N/2}$.

8.6.6 Plaintext message recovery via lattice attacks

Let $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$. In the following subsection, we figure out how big $\|\mathbf{e}\|$ has to be in order to prevent its recovery by lattice attacks. In our scheme, we do bit dropping, which virtually adds a huge error $(0, \mathbf{e}_3)$ to $\mathbf{e}$. We will however act like there is no bit dropping and will not take $(0, \mathbf{e}_3)$ into account : even though the error added is tremendous, it is asymmetric (it affects only the second half) and deterministic. These specificities make it unclear how much bit dropping hinder lattice attacks, therefore we will choose to ignore bit dropping, giving the adversary more information that it actually has.

In order to prevent recovery of the $\mathbf{e}$, we need $\frac{\sqrt{\frac{2qN}{2\pi e}}}{\|\mathbf{e}\|} < 0.4 \cdot \gamma^{2N}$. In practice, taking $\|\mathbf{e}\| \approx \sigma_r \sqrt{2N}$, with $\sigma_r = \frac{3q}{16\sqrt{2\lambda \log 2}\|\mathbf{s}\|}$ ensures correct decryption. This gives the (simplified) condition:

$$\eta'_\epsilon(\mathbb{Z})\sqrt{\lambda N} < 0.17\gamma^{2N}$$

8.6.7 Analysis of the signature scheme

In our signature scheme, the keygen is provided by Algorithm 8.1, the signature by Algorithm 8.2 and the verification by checking the norm of $(\mathbf{s}_1, \mathbf{s}_2)$ as well as the equality $\mathbf{s}_1 + \mathbf{s}_2 * \mathbf{h} = H(\mathbf{m})$. Since there is no encryption, we can discard the CPA-security analysis at the end of the previous section, as well as the issues regarding correctness of the encryption. This leads to much smaller values for N and q, which can be found in the Table 8.3 of Section 8.1.

We now analyze the bitsize of the private key, public key and signature. The public key is $\mathbf{h} \in \mathcal{R}_q$, as well as the signature $\mathbf{s}_1$, so their bitsizes are $N\lceil \log_2 q \rceil$. The private key is f such that $f * \mathbf{h} = g \bmod q$. Given the procedure to generate $\mathbf{b}_1 = (f, g)$, with high probability each coefficient of f has absolute value at most equal to $6\sigma_f$ (if it isn't the case, one just needs to re-sample the coefficient). f can therefore be stored using $N(1 + \lceil \log_2(6\sigma_f) \rceil)$ bits, where $\sigma_f = 1.17\sqrt{\frac{q}{2N}}$.

Using Huffman coding, as in BLISS [DDLL13], may also be appropriate here for reducing the key length by several hundred bits, but we do not expand on this idea.

8.7 Conclusion

Trapdoor sampling is at the heart of many "advanced" lattice constructions, yet it has not been previously considered to be viable in practice. In this thesis, we showed that with a

proper distribution on the trapdoor as well as analyzing the outputs using KL divergence instead of statistical distance, one can have schemes that are rather efficient and have their security based on the hardness of lattice problems over NTRU lattices. We believe that this opens the door to further practical implementations of lattice primitives having the GPV trapdoor sampling algorithm at their core.

Our work used a distribution over NTRU lattices that is somewhat new – rather than having very short vectors, our private key has vectors with a small Gram-Schmidt maximum. It is unclear how this compares in terms of difficulty to the hardness of lattice problems under the "standard" NTRU distribution. On the one hand, the vectors in our private key are longer, but on the other hand, our private key is more "orthogonal". General lattice algorithms (such as BKZ) don't seem to exploit this feature, but it is an interesting open problem whether other techniques could be used for improved cryptanalysis of our schemes.

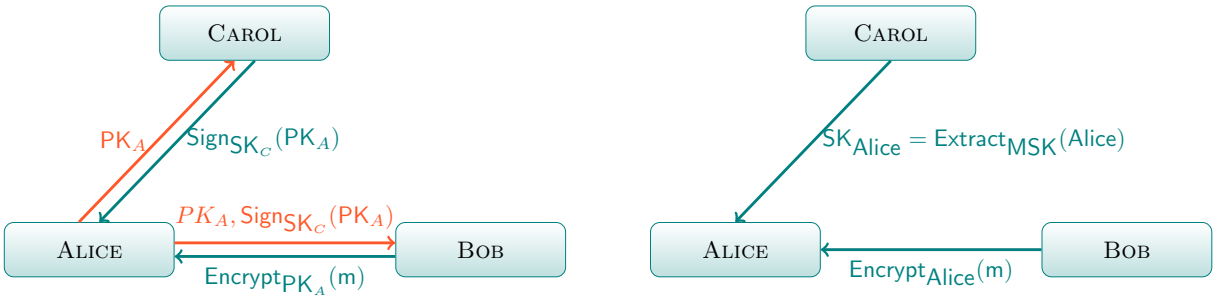
Chapter 9

Applications of Identity-Based Encryption

9.1 Introduction

In the previous chapter, we presented an efficient identity-based encryption (IBE) scheme based on lattices. In this chapter, we will take look at the bigger picture and consider its potential applications.

Figure 9.1 gives a simplified illustration of an identity-based infrastructure (on the right). The Setup and Decrypt steps are omitted for clarity and since they create no communication costs (except a single broadcast feed at the end of Setup). More importantly, Figure 9.1 compares the communication costs of a classical public key infrastructure (PKI) versus an identity-based infrastructure in the simple two-users setting, and the latter boasts reduced communication costs compared to the former.



A Classical PKI

An identity-based PKI

Figure 9.1: IBE PKI versus Classical PKI

From the model we described, we can see three immediate properties inherent to identity-based encryption, which are also clear in Figure 9.1:

- The communication costs of an identity-based infrastructure are lower than for a classical PKI. This is not only a gain in communication, but also in security, as less communication means a lesser number of possible attacks (e.g. *man-in-the-middle* attacks).

- A user Alice doesn't have to send any information to receive an encrypted message. Two notable consequences are that Alice doesn't need any outward communication ability, and Bob can send a message to Alice before an account is even created for her on an IBE infrastructure.
- The authority (Carol in the figure) holds in escrow all the user keys it generated. This property may be viewed as undesirable (from a user's point of view), desirable (for the authority) or neutral (e.g. when the user is a non-sentient entity) and contextual factors may also influence the desirability of key escrow.

Taking time as a parameter also unlocks interesting features such as revocation [BF01, BGK08] and blocking decryption until a fixed date.

9.2 Authenticated Key Exchange in a Constrained Environment

In many situations (more thoroughly described in Section 9.3), it is desirable to exchange keys in a non-interactive way. This can be achieved by the notion of ID-based non-interactive key exchange.

Definition 9.1 (ID-NIKE)**.** *Reprising the setting and notations of Section 8.2.1, we define an ID-based non-interactive key exchange (ID-NIKE) by three distinct algorithms:*

- Setup *and* $\mathsf{Extract}$ *are identical to their counterparts of Section 8.2.1.*
- $\mathsf{Exchange}(\mathsf{MPK}, \mathsf{SK}_{\mathsf{id}_A}, \mathsf{id}_B)$*: Given a private key* $\mathsf{SK}_{\mathsf{id}_A}$ *associated to* id_A*, and* $\mathsf{id}_B \neq \mathsf{id}_A$*, outputs a key* $\mathsf{K}_{A,B}$*. We require that for identities* $\mathsf{id}_A \neq \mathsf{id}_B$ *and valid private keys* $\mathsf{SK}_{\mathsf{id}_A}, \mathsf{SK}_{\mathsf{id}_B}$*, we have the following equality:*

$$\mathsf{Exchange}(\mathsf{MPK}, \mathsf{SK}_{\mathsf{id}_A}, \mathsf{id}_B) = \mathsf{Exchange}(\mathsf{MPK}, \mathsf{SK}_{\mathsf{id}_B}, \mathsf{id}_A) \tag{9.1}$$

Equation 9.1 allows two users to share a session key in a non-interactive manner, knowing each other's identities and their own private keys. Pairing-based IBEs allow to construct ID-NIKE [SOK00, DE02, PS07].

While it would be great to achieve ID-NIKE from a lattice-based IBE, it is unclear at this point if one can achieve that with our scheme. One can generically construct an IBE from an ID-NIKE [PS09], but it is unknown if the converse is true.[1] Works by Ding et al. [DXL12], Fujioka et al. [FSXY13], Peikert [Pei14] and Bos et al. [BCNS15] have considered transposing Diffie and Hellman's key exchange [DH76] in the RLWE setting, resulting in some very efficient implementations [BCNS15, GAGL15]. However, following this idea with our IBE leads to the following (non-interactive) key-exchange:

$$\mathsf{Alice} \underset{H(\mathsf{Bob})=\mathbf{h}*\mathbf{s}'_1+\mathbf{s}'_2}{\overset{H(\mathsf{Alice})=\mathbf{h}*\mathbf{s}_1+\mathbf{s}_2}{\rightleftharpoons}} \mathsf{Bob}$$

where each coefficient of $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}'_1, \mathbf{s}'_2$ is expected to have absolute value about $\sigma > \sqrt{q}$, which is too big for Alice and Bob in the example above to share a secret $\mathsf{K} \approx \mathbf{s}_1 * \mathbf{h} * \mathbf{s}'_1$ as done in [Pei14, BCNS15].

[1] [PS09] suggests that it is not.

However, instead of ID-NIKE, we can achieve ID-based unidirectional authenticated key exchange, which we believe is powerful enough in many practical settings.

Definition 9.2 (ID-UAKE)**.** *Reprising the setting and notations of Section 8.2.1, we define an ID-based unidirectional authenticated key exchange (ID-UAKE) by four distinct algorithms (the last three implicitly take* MPK *as an argument):*

- Setup *and* $\mathsf{Extract}$ *are identical to their counterparts of Section 8.2.1.*
- $\mathsf{Send}(\mathsf{SK}_{\mathsf{id}_A}, \mathsf{id}_B, \mathsf{K})$*: Given a private key* $\mathsf{SK}_{\mathsf{id}_A}$ *associated to* id_A*, and* $\mathsf{id}_B \neq \mathsf{id}_A$*, outputs an encrypted message* C*.*
- $\mathsf{Recover}(\mathsf{SK}_{\mathsf{id}_B}, \mathsf{id}_A, \mathsf{C}))$*: Given* $(\mathsf{SK}_{\mathsf{id}_B}, \mathsf{id}_A)$ *and an encrypted message* C*, outputs* $\perp$ *or a key* K*.*

We require that for identities $\mathsf{id}_A \neq \mathsf{id}_B$ *and valid private keys* $\mathsf{SK}_{\mathsf{id}_A}, \mathsf{SK}_{\mathsf{id}_B}$*, we have (with overwhelming probability) the following equality:*

$$\mathsf{Recover}(\mathsf{SK}_{\mathsf{id}_B}, \mathsf{id}_A, \mathsf{Send}(\mathsf{SK}_{\mathsf{id}_A}, \mathsf{id}_B, \mathsf{K}))) = \mathsf{K}$$

This allows a user A *to send a key* K *to another user* B*, which they then share.*

Consider a scenario where Bob wants to securely send a session key K to Alice, but Alice is unable to communicate with Bob and none of them can communicate with the authority. Such a situation can occur to wireless sensor networks, email users (Section 9.3) or botnets (Section 9.4). Using IBE in conjunction with an identity-based signature scheme (IBS), Bob can send an encrypted and authenticated session key to Alice by following the protocol described in Figure 9.2.

- $\mathsf{Setup}(1^n)$: Generate an infrastructure for IBE as well as for IBS.
- $\mathsf{Extract}(\mathsf{id})$: Extract a private key $\mathsf{SK}_{\mathsf{id}}$ which contains a decryption key for the IBE and a signing key for the IBS (so $\mathsf{SK}_{\mathsf{id}}$ can be used either with Sign or Verify).
- $\mathsf{Send}(\mathsf{SK}_{\mathsf{Bob}}, \mathsf{Alice}, \mathsf{K})$: If Bob wants to send to Alice a session key K authenticated under his name and readable by Alice alone, he does the following:

  1. $\mathsf{S} \leftarrow \mathsf{Sign}_{\mathsf{SK}_{\mathsf{Bob}}}(\mathsf{Alice} \| \mathsf{K})$
  2. $\mathsf{C} \leftarrow \mathsf{Encrypt}_{\mathsf{Alice}}(\mathsf{K} \| \mathsf{S})$
  3. Output C

- $\mathsf{Recover}(\mathsf{SK}_{\mathsf{Alice}}, \mathsf{Bob}, \mathsf{C}))$: To authenticate and recover an encrypted session key K sent by Bob, Alice does the following:

  1. $(\mathsf{K}' \| \mathsf{S}') \leftarrow \mathsf{Decrypt}_{\mathsf{SK}_{\mathsf{Alice}}}(\mathsf{C})$
  2. $b \leftarrow \mathsf{Verify}_{\mathsf{Bob}} \left[\mathsf{S}' = \mathsf{Sign}_{\mathsf{SK}_{\mathsf{Bob}}}(\mathsf{Alice} \| \mathsf{K}')?\right]$
  3. If $b = 1$, output K'. Else, output $\perp$.

Figure 9.2: An ID-based unidirectional authenticated key-exchange

The UAKE described in Figure 9.2 follows an ID-based Sign-then-Encrypt paradigm.[2] We do not provide a formal security analysis, however, provided that the IBE used has indistinguishability of ciphertexts under adaptive chosen-ciphertext attacks (IND-CCA2) and that the IBS used is existentially unforgeable under chosen-message attacks (UF-CMA), its security follows from Theorem 1 of [ADR02]. This analysis seamlessly transfers to the ID-based setting [Boy10a].

A generic and simple transformation to turn any signature scheme into an IBS is for the signer to be provided by the authority with a signing key and a certificate for the associated verification key, and to append the verification key and certificate to the signature he sends to Alice. More details about this transformation – and others more efficient – can be found in [KN09]. For complete post-quantum resilience, we suggest taking certificated BLISS [DDLL13] as the IBS scheme.

There exist more efficient constructions of key-exchange from IBE and IBS [Boy10a, DZ10] or of IBS from signature schemes [KN09], but we voluntarily settled with a "naive" approach to show the feasibility of UAKE from the available lattice-based primitives.

In the two next sections, we give a few applications of IBE and indicate use cases for which we believe our ID-UAKE can accomplish the same functionalities as would an ID-NIKE.

9.3 Selected Applications of Identity-Based Encryption

We now give a few applications of identity-based encryption. As explained before, even without ID-NIKE, identity-based encryption can solve interesting challenges for any of these applications as ID-UAKE is sufficient for the use cases we will mention.

9.3.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are – the denomination is rather transparent – networks of wireless sensors able to gather and transmit environmental data in an autonomous way. They have numerous applications: most notable application include using them to monitor toxic gases or radioactivity level in cities or industrial sites, detect natural disasters, survey military sensible areas, monitor tire pressure (TPMS) and for health care.

From a cryptographic point of view, the main challenge of WSNs is to provide a safe cryptographic environment to devices whose computational, storage, power and communication abilities may be very limited. Identity-based encryption has stirred a wide interest in the WSN community for its ability to elegantly solve non interactive key distribution and key-exchange problems [SOK00]. Several constructions have been proposed [YRV+06, ODL+07, GRL08, SC09, CLZ+10, PS12a], as well as implementations of pairing computations [OAG+11, OAM+07, SC09, XWD10]

Many WSNs have a multi-hop mesh network topology. In this case, communication between the authority and a node, or between two nodes of the WSN can be uncertain. In addition, wireless communication is energy consuming and nodes typically have very limited power or may even be devoid of transmission abilities. With our scheme, a node that

[2] We in fact use a slightly hardened Sign-then-Encrypt because of the vulnerability of the original paradigm to "surreptitious forwarding" [Dav01]. That being said, the attack in the aforementioned paper applies to RSA-like schemes and it is unclear whether it applies in our case.

is energy-limited and/or isolated from the authority can receive authenticated encrypted messages from another node. Very efficient hardware implementations of the encryption and decryption parts already exist, speed-wise [RVM+14, LSR+15, POG15, CMV+15] and area-wise [PG14a]. Since in our scheme the encryption and decryption are much faster than their pairing-based counterparts, we believe that using IBE in WSNs would benefit from our IBE (or similar lattice-based IBEs).

9.3.2 The Voltage SecureMail™ Service

The first industrial application of IBE was proposed by Voltage Security, a company co-founded by Dan Boneh. This security solution, SecureMail™, provides its users with an encrypted mail service using Identity-Based Encryption.

A main selling point of this service is its simplicity: official documents by Voltage Security [Sec, Spi04] state it can provide companies with a key management system that is much lighter than usual public-key infrastructures and can be easily integrated in current architectures.

While it is hard to find independent evaluations of Voltage's IBE solution, it is worth noting that Hewlett-Packard has currently acquired Voltage Security and currently propose Identity-Based Encryption as a security feature[3].

As an application of our scheme, a customer using the Voltage SecureMail™ could send a mail to another user even if the authority server is down.

9.4 Botnets

The recent years have seen a surge of botnets, which are networks of virus-infected computers who obey to a distant server. The spread of botnets is a very recent and still-changing phenomenon. Out of the many definitions, we chose the one given by Rajab, Zarfoss, Monrose and Terzis [RZMT06]:

> The term botnets is used to define networks of infected end-hosts, called bots, that are under the control of a human operator commonly known as the botmaster. While botnets recruit vulnerable machines using methods also utilized by other classes of malware (e.g., remotely exploiting software vulnerabilities, social engineering, etc.), their defining characteristic is the use of command and control (C&C) channels to connect bots to their botmasters.

We briefly recall the lifecycle of botnets, which follows essentially four phases:

1. **Spread.** A bot operator – or botmaster – sends out viruses and worms on the internet to infect computers.

2. **Infect.** Those viruses infect computers who then become partially or totally controlled by bots.

3. **Command & Control.** The bot in an infected computer connects to a server – often called C&C server – controlled by the botmaster to receive instructions.

[3] https://www.voltage.com/technology/data-encryption/identity-based-encryption/

4. **Attack.** The botmaster can issue any order to the bots connected.

Readers interested in more details may read [SSPS13] as an introduction. Frequent uses of botnets include distributed denial-of-service (DDoS) attacks, spamdexing (*ie* spamming requests over a web search engine in order to influence the ranking of a site), network scanning, email address harvesting (bots doing so are called *spambots*) bitcoin mining and the theft of valuable information such as credit card numbers, login ID's and application serial numbers.

Botnets' illegal nature give them some properties that are desirable to achieve:

- **Stealth.** The bots must be stealthy. To avoid detection by an antivirus or an intrusion detection system (IDS), it is preferable to keep the storage, energy and bandwidth consumption low whenever possible.

- **Autonomy.** The ability for a botnet to autonomously maintain its activities without communicating with a C&C server is desirable in at least two scenarios:

  1. A C&C server is identified and taken down.
  2. Part or totality of the bots is suddenly isolated from the internet. This may happen when a local network is quarantined under discovery or suspicion of infection.

  We stress that the two scenarios given above happen in real life on a monthly (resp. daily) basis. Recent botnets have adapted to the first scenario by using peer-to-peer networks [ARSG+13].

  This (semi-)autonomy requirement can be useful in particular for botnets aiming at DDoS attacks, spamdexing, network scanning, email harvesting and bitcoin mining, since these activities can be performed more efficiently when participants coordinate their actions.

- **Resilience.** Compromise of a single bot should leave the rest of the botnet unharmed.

9.4.1 Botnets and Encryption

Malware developers have always been enthusiastic users of encryption, and botnet developers are no exception. The first motivation was to encrypt (parts of) the program to hide its maliciousness and to prevent its detection and reverse-engineering as much as possible. Popular encryption algorithms feature TEA[WN95] (Storm, SilentBanker [CPKS09]), AES [AES01] or RC4 (Sality [Zak10]). As a result, reverse-engineering malwares to recover cryptographic keys and/or primitives has become an active area of research [LMW09, GWH11, CFM12].

More recently, a threat originally envisioned by Young and Yung [YY96] under the name of cryptoviral extortion became in a few years a significant part the cybercrime landscape under the name of *ransomwares*. Upon infection of a computer, ransomwares proceed to encrypt its whole content and to erase the original content along with the encryption key. Without the key, the user of the infected computer has no way to recover its data, unless he or she pays a ransom to the operator of the ransomware.

Ransomwares have been witnessed to use more robust cryptographic schemes than other malwares, such as AES-256 alone (Teslacrypt[4] [ACT15]), or in conjunction with

[4]Teslacrypt claimed to rely on RSA-2048 but was discovered to use only AES-256. As a result, it was reverse-engineered by researchers from Cisco and subsequently dismantled [ACT15].

RSA-2048 [RSA78] (Cryptowall, Cryptolocker [KW14], Torrentlocker [Lév14]). Two recent and seemingly independent ransomwares, OphionLocker and Critroni (a.k.a. CTB-Locker – for Curve-Tor-Bitcoin) even use elliptic-curve cryptography to encrypt the infected users' files [Sch14].

For C&C communication, encryption is less sophisticated: most botnets either use no encryption, the RC4 cipher (Cryptowall, Zeroaccess [Mor12]) or (variants of) XOR encryption [RD13, PSY]. However, recent botnets have been using stronger cryptography on this aspect too. Torrentlocker uses SSL [Lév14], Waledac uses AES to encrypt its C&C traffic [CDB09] and Nugache uses variable-length RSA key exchange with Rijndael-256 [DR02] session keys for each peer connection.

9.4.2 Identity-Based Encryption for Botnets

Botnets face analog challenges to WSNs. They need to maintain a cryptographically secure infrastructure with low storage, power and bandwidth abilities. In addition, the topology of the network may change due when part of the botnet is discovered and taken down. We review the possibility of relying on "basic" cryptography:

1. Relying on symmetric key cryptography alone is insecure. A captured botnet can be reverse-engineered and the key exposed – see Section 9.4.1 and footnote 4.

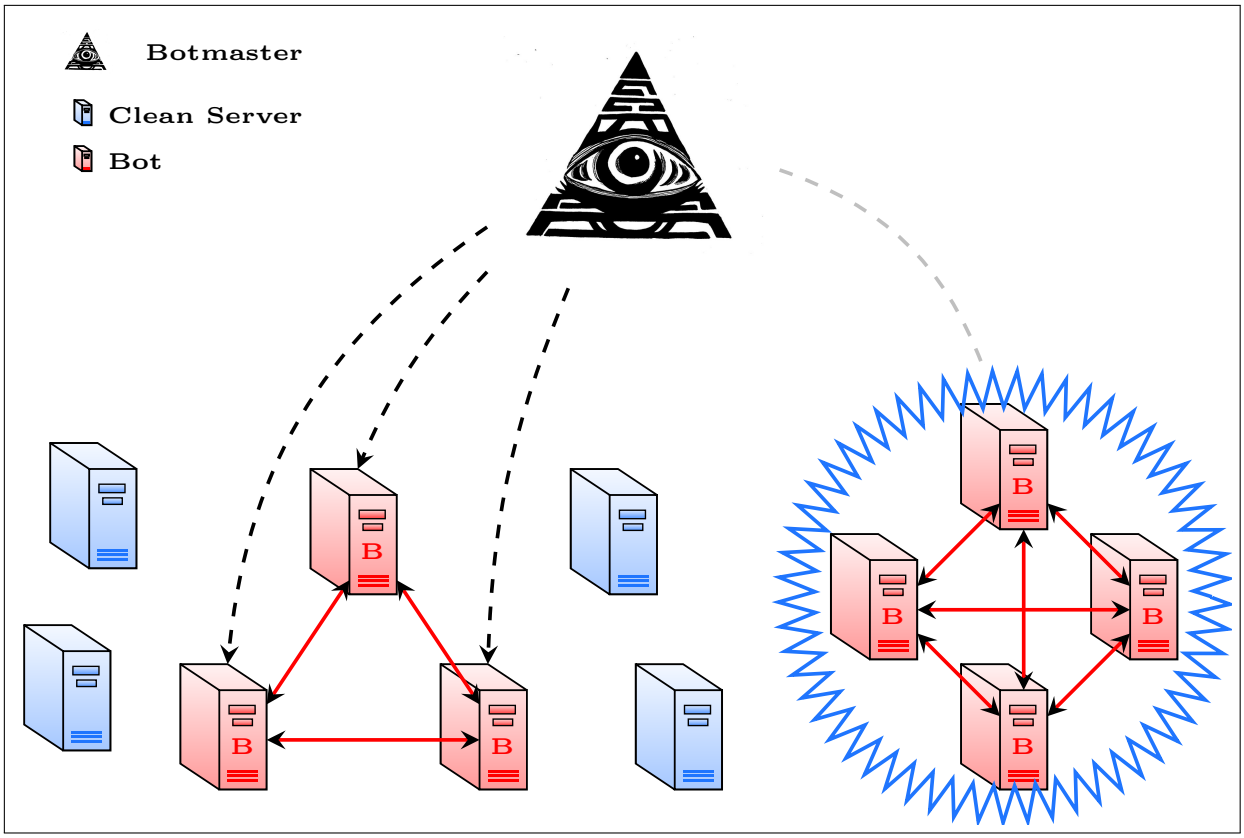
2. Each bot may have its own certified public key, and store its private key. This works most of the times – but not all the time. We explain why below.

The illegal nature of botnets make their environment fundamentally hostile. The botmaster may be taken down or part of the network may be isolated. For this reason, botnets have moved from centralized architectures to mesh networks or peer-to-peer networks. Recent botnets even use social networks to obtain C&C instructions. In these configurations, a bot may have to send a message to another one without the latter being able or willing to answer – because they are too many hops away or because they don't want suspicious network activity to be discovered.

This places us exactly in the situation that we described in Section 9.2. In this setting, an IBE can provide secure authenticated key exchange with minimal communication. In addition, some properties specific to *lattice-based* IBEs may be desirable:

1. In pairing-based IBE, encryption and decryption are costly and rely on multi-precision arithmetic. These operations are done by the bots and may make them easier to detect. On the other hand, the encryption and decryption operations in our lattice-based IBE are very fast and use only standard, small-precision arithmetic.

2. Our IBE is anonymous, which is a very useful feature for botnets. We note that is is also the case of some other IBEs [BW06].



The botmaster gives private keys and orders to each bot *via* a channel that might be interrupted (symbolized by ⇢). The bots can securely send messages to each other even if part of the botnet is suddenly isolated (blue broken line circle) from the rest of the network.

Figure 9.3: A botnet using IBE.

9.5 Conclusion

We have presented a few applications of identity-based encryption. ID-based non-interactive key exchange as done in pairing-based IBEs [SOK00, DE02, PS07] is very useful, but for many of the considered applications, a slightly relaxed form of it suffices, and it can be provided with our lattice-based IBE. In the future years, many of the aforementioned applications are very likely to become widespread, especially wireless sensor networks and botnets. Identity-based protocols will be very useful then, and the efficiency of our lattice-based encryption scheme may ease and accelerate the adoption of such protocols.

Part IV

Conclusion

Practical Lattice-Based Cryptography. For anyone interested in bringing lattice-based cryptography closer to "the real world", the last three years have proved to be particularly exciting. New efficient schemes have been proposed for signature [DDLL13] and authenticated key exchange [Pei14, BCNS15], along with (open-source) implementations that confirm their efficiency [DDLL13, BCNS15, GAGL15]. In addition, hardware implementation have shown that this efficiency transfers to constrained devices.

Identity-based encryption is certainly not a primitive as vital as the two mentioned above (as well as simple encryption), but its importance may grow stronger in the future. It is already seriously considered to be implemented in wireless sensor networks – see Section 9.3.1 – and such networks, as well as all the other systems encompassed by the umbrella term *Internet of Things* are expected to significantly develop in the years to come. With hacking of wireless sensors already being demonstrated as a way to disrupt cars, drones, pacemakers and even rifles, securing sensor networks already appear of vital importance. And this security issue will become even more acute if human bodies start to become embedded with wireless sensors and connected limbs, which I believe will happen in a few decades' time. Lattice-based identity-based encryption may then play a major role, not only because it is identity-based, but because the efficiency and post-quantum resilience of lattices will play in their favor when compared to pairing-based cryptography.

Hardware Implementation. The main immediate selling point of lattice-based cryptography at this current time may not be its post-quantum resistance, but its speed efficiency. Unlike RSA-like and elliptic-curve cryptography, there is no costly operation such as exponentiation. In addition, for many schemes no multi-precision integers are needed outside of the key generation, which once again stands in stark contrast with the classical schemes aforementioned. For these reasons, lattice-based cryptography seems like an ideal candidate for implementation on embedded devices.

And indeed, these last few years have seen a growing enthusiasm of the cryptographic hardware community towards lattices, which has led to recent efficient implementations of lattice-based schemes [PG12, GLP12, PG14b, DBG+15, PG14a, RVM+14, POG15, CMV+15], including the NTRU encryption scheme [BCE+01, ABF+08, KY09] and the BLISS signature scheme [PDG14]. I believe that lattice-based cryptography is now ready for the next logical step to its deployment in real life, that might also prove one of the most challenging ones. This step is withstanding fault attacks.

A large part of lattice-based cryptography – including hardware implementations – relies on discrete Gaussians and/or rejection sampling. Compromising one of these two steps could prove devastating to unprotected lattice-based cryptography. As an example, consider an output $\sum_i z_i \mathbf{b}_i$ of a Gaussian sampler (that we consider centered on 0 for clarity). The z_i's are supposed to be distributed according to a Gaussian, but a fault attack that makes them uniform in an interval would open the door to attacks in the line of [NR06, DN12b] and expose the private basis. Similar attacks would apply to unprotected BLISS, and would be extremely powerful as an attacker would not need to flip or control one single bit, but to randomize an array of bits, which is generally easier for fault attacks.

Post-Snowden Cryptography. A scheme is only as secure as the assumptions it lays upon. The bombshell revelations made by Edward Snowden in 2013 have shown a somewhat surprising ability of the NSA to circumvent many of these assumptions. Standout examples include tampering with standard bodies to promote weak cryptography, implementing backdoors where no one can detect them or subpoenaing firms into handling them keys or valuable data.

From a cryptographer's perspective, I think it should push us out of our comfort zone and consider that from the design of a cryptographic scheme to the context (user, provider, country, machine, etc.) of its use in real life, any step can potentially be compromised.

In the field of this thesis, it raises some intriguing questions: is there a way to introduce a backdoor in a lattice-based cryptographic scheme? Can a logical or hardware pseudo-random number generator (or a Gaussian generator!) be altered in a way that a lattice-based scheme using it would be insecure unbeknownst to its users?

A governmental entity can also do a *quantum bet*: collect and store as much pre-quantumly encrypted information as possible, and decrypt it when a quantum computer is available. It would then be in possession of information ranging over as many years (or even decades) as its target neglected to move to cryptography resilient to quantum computers. If we look at the NSA, it released an official statement recommending its partners to move to post-quantum cryptography [NSA15] and has invested $79.7 million to research the possible realization of a quantum computer[RG14]. Moreover, it is currently building a data center in Utah that is speculated to be able to store 50 times the current yearly contents of all voice communications in the United States [Hil13]. Given all this, there seems to be no is no technical reason that would prevent the NSA, or an equivalently powerful entity, to do such a bet.

Cryptanalysis. This is maybe the least understood part of lattice-based cryptology now, which is very unfortunate in my opinion. Lattices are a powerful tool that allows to create a wide range of cryptographic primitive, but many proposed schemes don't feature concrete parameters, which from a cryptanalyst's standpoint is not very attractive. This is why I strongly support the papers that propose practical parameters: by giving the community specific challenges to attack, they help making lattice-based cryptography safer.

For this same reason, I have to mention the works of Chen, Gama and Nguyen [GN08, CN11] which helped lattice practitioners to have a better understanding of the hardness of lattice problems. State-of-the-art reports such as the ones of Laarhoven, van de Pol and de Weger [LvdPdW12] about practical security of lattice-based cryptosystems, and of Albrecht, Player and Scott [APS15] about LWE are also very valuable tools for the whole community. And of course the recent efforts [CHL+15, FLLT15] to break lattice-based schemes play an essential role.

Quantum Computers. When I started my PhD, I often heard post-quantum resilience as the main selling point of lattice-based cryptography compared to classical cryptography. Quantum computers would wipe away the old number-theoretic schemes and post-quantum cryptography – such as lattice-based – would take their place. But quantum computers don't only promise to bring the long-announced cryptocalypse that may break most of the public-key cryptosystems we currently use. Their fundamentally different nature is supposed to revolutionize the treatment of many difficult tasks. However, promises are what they are and the advent of quantum computers could be stopped by many factors – including the law of physics. Nietzsche said:

> There are two different types of people in the world, those who want to know, and those who want to believe.

I enjoyed these three years playing with lattices, independently of the existence of quantum computers. And I cannot just take their advent for granted. For this reason, I would like to see a practical quantum computer in two decades from now.

Bibliography

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572.
→ Cited on page 41.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Rabin [Rab10], pages 98–115.
→ Cited on pages 17, 41, 61, 90, and 137.

[ABDG14] Carlos Aguilar Melchor, Xavier Boyen, Jean-Christophe Deneuville, and Philippe Gaborit. Sealing the leak on classical NTRU signatures. Cryptology ePrint Archive, Report 2014/484, 2014. http://eprint.iacr.org/2014/484.
→ Cited on page 101.

[ABF+08] Ali Can Atici, Lejla Batina, Junfeng Fan, Ingrid Verbauwhede, and Siddika Berna Örs. Low-cost implementations of NTRU for pervasive security. In *19th IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2008, July 2-4, 2008, Leuven, Belgium*, pages 79–84. IEEE Computer Society, 2008.
→ Cited on page 164.

[ABFK14] C. Aguilar, J. Barrier, L. Fousse, and M.O. Killijian. Xpire : Private information retrieval for everyone. 2014.
→ Cited on page 138.

[ACT15] Andrea Allievi, Earl Carter, and Emmanuel Tacheau. Threat spotlight: Teslacrypt – decrypt it yourself, 2015. http://blogs.cisco.com/security/talos/teslacrypt.
→ Cited on page 160.

[ADM12] Michel Abdalla, Angelo De Caro, and Karina Mochetti. Lattice-based hierarchical inner product encryption. In Hevia and Neven [HN12], pages 121–138.
→ Cited on page 90.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Knudsen [Knu02], pages 83–107.
→ Cited on page 158.

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 733–742, 2015.
→ Cited on pages 17 and 42.

[ADS15] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the closest vector problem in 2^n time - the discrete gaussian strikes again! *CoRR*, abs/1504.01995, 2015.
→ Cited on pages 17 and 42.

[AES01] Advanced encryption standard (aes). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.
→ Cited on page 160.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Lee and Wang [LW11], pages 21–40.
→ Cited on page 90.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.
→ Cited on page 143.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.
→ Cited on pages 16, 21, 99, and 116.

[AP08] Joel Alwen and Chris Peikert. Generating shorter bases for hard random lattices. Cryptology ePrint Archive, Report 2008/521, 2008. http://eprint.iacr.org/2008/521.
→ Cited on pages 137 and 139.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.
→ Cited on page 116.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. http://eprint.iacr.org/2015/046.
→ Cited on page 165.

[Arn51] Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1–2):17–29, 1951.
→ Cited on page 64.

[ARSG+13] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos. Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus. In *Malicious and Unwanted Software: "The Americas" (MALWARE), 2013 8th International Conference on*, pages 116–123, Oct 2013.
→ Cited on page 160.

[Bab85] L Babai. On lova´sz' lattice reduction and the nearest lattice point problem. In *Proceedings on STACS 85 2Nd Annual Symposium on Theoretical Aspects of Computer Science*, pages 13–20, New York, NY, USA, 1985.
→ Cited on pages 24, 29, 30, 61, 62, 85, 94, and 113.

[Bab86] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
→ Cited on pages 24, 29, 30, 61, 62, 85, 94, and 113.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.
→ Cited on page 137.

[BC84] G. R. Blakley and David Chaum, editors. *CRYPTO'84*, volume 196 of *LNCS*. Springer, Heidelberg, August 1984.
→ Cited on pages 171 and 179.

[BCE+01] Daniel V. Bailey, Daniel Coffin, Adam J. Elbirt, Joseph H. Silverman, and Adam D. Woodbury. NTRU in constrained devices. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 262–272. Springer, Heidelberg, May 2001.
→ Cited on page 164.

[BCG+14] Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. Discrete ziggurat: A time-memory trade-off for sampling from a gaussian distribution over the integers. In Lange et al. [LLL14], pages 402–417.
→ Cited on pages 18, 57, and 116.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570. IEEE Computer Society, 2015.
→ Cited on pages 156 and 164.

[Ber14] Dan Bernstein. A subfield-logarithm attack against ideal lattices. http://blog.cr.yp.to/20140213-ideal.html, Febuary 2014.
→ Cited on pages 81 and 86.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.
→ Cited on pages 135, 137, 142, and 156.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Nguyen and Oswald [NO14], pages 533–556.
→ Cited on pages 17, 41, 61, and 137.

[BGK08] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08*, pages 417–426. ACM Press, October 2008.
→ Cited on page 156.

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. ASIACRYPT, 2015.
→ Cited on pages 47, 114, 116, and 147.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Boneh et al. [BRF13], pages 575–584.
→ Cited on pages 18, 21, 42, 87, 114, and 116.

[BM58] G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29(2):610–611, 06 1958.
→ Cited on page 57.

[Boy10a] Xavier Boyen. Identity-based signcryption. In Dent and Zheng [DZ10], pages 195–216.
→ Cited on page 158.

[Boy10b] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, Heidelberg, May 2010.
→ Cited on pages 17, 41, 61, and 137.

[Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 122–142. Springer, Heidelberg, March 2013.
→ Cited on pages 41, 113, and 116.

[BR14] Lejla Batina and Matthew Robshaw, editors. *CHES 2014*, volume 8731 of *LNCS*. Springer, Heidelberg, September 2014.
→ Cited on pages 177 and 178.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *45th ACM STOC*. ACM Press, June 2013.
→ Cited on pages 169 and 173.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, Heidelberg, August 2006.
→ Cited on page 161.

[Cac97] Christian Cachin. *Entropy Measures and Unconditional Security in Cryptography*. PhD thesis, 1997.
→ Cited on page 46.

[CDB09] Joan Calvet, Carlton R. Davis, and Pierre-Marc Bureau. Malware authors don't learn, and that's good! In *4th International Conference on Malicious and Unwanted Software, MALWARE 2009, Montréal, Quebec, Canada, October 13-14, 2009*, pages 88–97. IEEE, 2009.
→ Cited on page 161.

[CDPR15] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. Cryptology ePrint Archive, Report 2015/313, 2015. http://eprint.iacr.org/.
→ Cited on pages 81 and 86.

[CFM12] Joan Calvet, José M. Fernandez, and Jean-Yves Marion. Aligot: cryptographic function identification in obfuscated binary programs. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 169–182. ACM Press, October 2012.
→ Cited on page 160.

[CG13] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part I*, volume 8042 of *LNCS*. Springer, Heidelberg, August 2013.
→ Cited on pages 170 and 176.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552.
→ Cited on pages 17, 40, 41, 61, 113, 116, and 137.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Oswald and Fischlin [OF15], pages 3–12.
→ Cited on page 165.

[CLZ+10] Cheng-Kang Chu, Joseph K. Liu, Jianying Zhou, Feng Bao, and Robert H. Deng. Practical ID-based encryption for wireless sensor network (short paper). In Dengguo Feng, David A. Basin, and Peng Liu, editors, *ASIACCS 10*, pages 337–340. ACM Press, April 2010.
→ Cited on page 158.

[CMV+15] Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray C. C. Cheung, Derek Pao, and Ingrid Verbauwhede. High-speed polynomial multiplication architecture for ring-lwe and SHE cryptosystems. *IEEE Trans. on Circuits and Systems*, 62-I(1):157–166, 2015.
→ Cited on pages 159 and 164.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Lee and Wang [LW11], pages 1–20.
→ Cited on pages 103 and 165.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, December 17–19, 2001.
→ Cited on page 135.

[CPKS09] Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Xiaodong Song. Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09*, pages 621–634. ACM Press, November 2009.
→ Cited on page 160.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Kaliski Jr. [Kal97], pages 410–424.
→ Cited on page 137.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
→ Cited on pages 113, 115, 117, and 121.

[CT91] Thomas M. Cover and Joy Thomas. *Elements of Information Theory*. Wiley, 1991.
→ Cited on page 47.

[Dav01] Don Davis. Defective sign & encrypt in s/mime, pkcs#7, moss, pem, pgp, and XML. In Yoonho Park, editor, *Proceedings of the General Track: 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Massachusetts, USA*, pages 65–78. USENIX, 2001.
→ Cited on page 158.

[DBG+15] Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sánchez, and Peter Schwabe. High-speed signatures from standard lattices. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 84–103. Springer, Heidelberg, September 2015.
→ Cited on page 164.

[DD12] Léo Ducas and Alain Durmus. Ring-lwe in polynomial rings. In *Public Key Cryptography–PKC 2012*, pages 34–51. 2012.
→ Cited on page 86.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Canetti and Garay [CG13], pages 40–56.
→ Cited on pages 11, 18, 47, 57, 61, 76, 83, 99, 101, 103, 116, 137, 138, 141, 150, 152, 158, and 164.

[DE02] Régis Dupont and Andreas Enge. Practical non-interactive key distribution based on pairings. Cryptology ePrint Archive, Report 2002/136, 2002. http://eprint.iacr.org/2002/136.
→ Cited on pages 156 and 162.

[DG14] Nagarjun C. Dwarakanath and Steven D. Galbraith. Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Appl. Algebra Eng. Commun. Comput.*, 25(3):159–180, 2014.
→ Cited on pages 18, 57, and 116.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
→ Cited on pages 15 and 156.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Heidelberg, December 2014.
→ Cited on pages 43, 61, 83, 84, 91, 114, 116, and 136.

[DMQ13] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In Johansson and Nguyen [JN13], pages 18–34.
→ Cited on page 47.

[DN12a] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In Wang and Sako [WS12], pages 415–432.
→ Cited on pages 18, 34, 46, 50, 57, 87, 92, and 116.

[DN12b] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Wang and Sako [WS12], pages 433–450.
→ Cited on pages 30, 34, 101, 143, and 164.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Secaucus, NJ, USA, 2002.
→ Cited on page 161.

[Duc13] Léo Ducas. *Lattice Based Signatures: Attacks, Analysis and Optimization*. Phd thesis, Ecole Normale Supérieure de Paris - ENS Paris, 2013.
→ Cited on pages 18, 57, and 116.

[Dur60] J. Durbin. The fitting of time-series models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 28(3):pp. 233–244, 1960.
→ Cited on pages 64 and 70.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. http://eprint.iacr.org/2012/688.
→ Cited on page 156.

[DZ10] Alexander W. Dent and Yuliang Zheng. *Practical Signcryption*. New York, NY, USA, 1st edition, 2010.
→ Cited on pages 158 and 169.

[EGM90] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital schemes. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 263–275. Springer, Heidelberg, August 1990.
→ Cited on page 36.

[eHGH+08] William Whyte (editor), Nick Howgrave-Graham, Jeff Hoffstein, Jill PIpher, Joseph H. Silverman, and Phil Hirschhorn. IEEE p1363.1 draft 10: Draft standard for public key cryptographic techniques based on hard problems over lattices. Cryptology ePrint Archive, Report 2008/361, 2008. http://eprint.iacr.org/2008/361.
→ Cited on page 101.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley and Chaum [BC84], pages 10–18.
→ Cited on page 16.

[FLLT15] Pierre-Alain Fouque, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of the co-acd assumption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 561–580, 2015.
→ Cited on page 165.

[FS10] Claus Fieker and Damien Stehlé. Short bases of lattices over number fields. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *Algorithmic Number Theory, 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings*, volume 6197 of *Lecture Notes in Computer Science*, pages 157–173, 2010.
→ Cited on page 94.

[FSXY13] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13*, pages 83–94. ACM Press, May 2013.
→ Cited on page 156.

[GAGL15] Adrien Guinet, Carlos Aguilar, Serge Guelton, and Tancrède Lepoint. Quatre millions d'échanges de clés par seconde. In *SSTIC 2015*, 2015.
→ Cited on pages 138, 156, and 164.

[Gal12] Steven D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, New York, NY, USA, 1st edition, 2012.
→ Cited on pages 38, 74, and 76.

[GFS+12] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin A. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In Prouff and Schaumont [PS12b], pages 512–529.
→ Cited on page 148.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Kaliski Jr. [Kal97], pages 112–131.
→ Cited on pages 30 and 37.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Johansson and Nguyen [JN13], pages 1–17.
→ Cited on page 47.

[GHN06] Nicolas Gama, Nick Howgrave-Graham, and Phong Q. Nguyen. Symplectic lattice reduction and NTRU. In Vaudenay [Vau06], pages 233–253.
→ Cited on pages 63, 64, 76, 77, 79, and 81.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May 2010.
→ Cited on pages 167, 170, and 175.

[GJSS01] Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocrypt 2001. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2001.
→ Cited on page 101.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Prouff and Schaumont [PS12b], pages 530–547.
→ Cited on page 164.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.
→ Cited on pages 103, 138, 139, and 165.

[Gol87] Oded Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 104–110. Springer, Heidelberg, August 1987.
→ Cited on page 38.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
→ Cited on pages 11, 17, 18, 31, 33, 34, 37, 38, 39, 42, 47, 50, 53, 61, 83, 85, 86, 91, 99, 101, 102, 113, 116, 135, 137, 139, 141, 143, and 147.

[Gra93] William B Gragg. Positive definite toeplitz matrices, the arnoldi process for isometric operators, and gaussian quadrature on the unit circle. *Journal of Computational and Applied Mathematics*, 46(1–2):183 – 198, 1993.
→ Cited on pages 64 and 70.

[GRL08] David Galindo, Rodrigo Roman, and Javier Lopez. A killer application for pairings: Authenticated key establishment in underwater wireless sensor networks. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *CANS 08*, volume 5339 of *LNCS*, pages 120–132. Springer, Heidelberg, December 2008.
→ Cited on page 158.

[GS66] W Morven Gentleman and Gordon Sande. Fast fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 563–578. ACM, 1966.
→ Cited on pages 113 and 116.

[GS02] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Knudsen [Knu02], pages 299–320.
→ Cited on page 101.

[Gui13] Aurore Guillevic. *Arithmetic of pairings on algebraic curves for cryptography*. Theses, Ecole Normale Supérieure de Paris - ENS Paris, December 2013.
→ Cited on page 139.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
→ Cited on page 65.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Boneh et al. [BRF13], pages 545–554.
→ Cited on pages 113 and 116.

[GWH11] Felix Gröbert, Carsten Willems, and Thorsten Holz. Automated identification of cryptographic primitives in binary programs. In Robin Sommer, Davide Balzarotti, and Gregor Maier, editors, *Recent Advances in Intrusion Detection - 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings*, volume 6961 of *Lecture Notes in Computer Science*, pages 41–60, 2011.
→ Cited on page 160.

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Heidelberg, August 2007.
→ Cited on page 150.

[HGHPW05] Nick Howgrave-Graham, Jeff Hoffstein, Jill Pipher, and William Whyte. On estimating the lattice security of NTRU. Cryptology ePrint Archive, Report 2005/104, 2005. http://eprint.iacr.org/2005/104.
→ Cited on page 151.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Heidelberg, April 2003.
→ Cited on pages 30, 37, 99, 100, 101, and 110.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002.
→ Cited on pages 58 and 74.

[Hil13] Kashmir Hill. Blueprints of NSA's ridiculously expensive data center in utah suggest it holds less info than thought. http://www.forbes.com/sites/kashmirhill/2013/07/24/blueprints-of-nsa-data-center-in-utah-suggest-its-storage-capacity-is-less-impressive-than-thought/, 2013.
→ Cited on page 165.

[HJB84] Michael T Heideman, Don H Johnson, and C Sidney Burrus. Gauss and the history of the fast fourier transform. *ASSP Magazine, IEEE*, 1(4):14–21, 1984.
→ Cited on page 113.

[HN12] Alejandro Hevia and Gregory Neven, editors. *LATINCRYPT 2012*, volume 7533 of *LNCS*. Springer, Heidelberg, October 2012.
→ Cited on pages 167 and 177.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 267–288, 1998.
→ Cited on pages 11, 16, 61, 76, 99, 101, 113, 116, 137, 138, 143, and 146.

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 211–228. Springer, Heidelberg, May 2001.
→ Cited on page 101.

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *EUROCRYPT 2013*, volume 7881 of *LNCS*. Springer, Heidelberg, May 2013.
→ Cited on pages 171, 172, and 175.

[Kal97] Burton S. Kaliski Jr., editor. *CRYPTO'97*, volume 1294 of *LNCS*. Springer, Heidelberg, August 1997.
→ Cited on pages 170 and 172.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, August 1987.
→ Cited on page 38.

[Kar13] Charles FF Karney. Sampling exactly from the normal distribution. *arXiv preprint arXiv:1303.6257*, 2013.
→ Cited on page 116.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 1883.
→ Cited on page 15.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941, 2000.
→ Cited on pages 18, 33, 42, 61, 85, 113, 116, and 137.

[KN09] Eike Kiltz and Gregory Neven. Identity-based signatures, 2009.
→ Cited on page 158.

[Knu02] Lars R. Knudsen, editor. *EUROCRYPT 2002*, volume 2332 of *LNCS*. Springer, Heidelberg, April / May 2002.
→ Cited on pages 167 and 172.

[Kob89] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, 1989.
→ Cited on page 16.

[KW14] Uttang Dawda Kyle Wilhoit. Your locker of information for cryptolocker decryption, 2014. https://www.fireeye.com/blog/executive-perspective/2014/08/your-locker-of-information-for-cryptolocker-decryption.html.
→ Cited on page 161.

[KY09] A.A. Kamal and A.M. Youssef. An fpga implementation of the ntruencrypt cryptosystem. In *Microelectronics (ICM), 2009 International Conference on*, pages 209–212, Dec 2009.
→ Cited on page 164.

[KYH+06] Takashi Kitagawa, Peng Yang, Goichiro Hanaoka, Rui Zhang, Hajime Watanabe, Kanta Matsuura, and Hideki Imai. Generic transforms to acquire CCA-security for identity based encryption: The cases of fopkc and REACT. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *Information Security and Privacy, 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006, Proceedings*, volume 4058 of *Lecture Notes in Computer Science*, pages 348–359, 2006.
→ Cited on page 147.

[Lan50] Cornelius Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral, 1950.
→ Cited on page 64.

[Lan14] Adeline Langlois. *Lattice - Based Cryptography - Security Foundations and Constructions*. Theses, Ecole normale supérieure de lyon - ENS LYON, October 2014.
→ Cited on page 47.

[Len82] Arjen K. Lenstra. Lattices and factorization of polynomials over algebraic number fields. In Jacques Calmet, editor, *Computer Algebra, EUROCAM '82, European Computer Algebra Conference, Marseille, France, 5-7 April, 1982, Proceedings*, volume 144 of *Lecture Notes in Computer Science*, pages 32–39, 1982.
→ Cited on pages 24 and 113.

[Lep14] Tancrède Lepoint. *Design and Implementation of Lattice-Based Cryptography*. Theses, Ecole Normale Supérieure de Paris - ENS Paris, June 2014.
→ Cited on pages 18, 57, and 116.

[Lev47] Norman Levinson. The Wiener RMS (root mean square) error criterion in filter design and prediction. *J. Math. Phys. Mass. Inst. Tech.*, 25:261–278, 1947.
→ Cited on pages 43, 64, and 70.

[Lév14] Marc-Étienne M. Léveillé. Torrentlocker - ransomware in a country near you, 2014. http://www.welivesecurity.com/wp-content/uploads/2014/12/torrent_locker.pdf.
→ Cited on page 161.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
→ Cited on pages 24, 29, 65, 74, 113, and 116.

[LLL14] Tanja Lange, Kristin Lauter, and Petr Lisonek, editors. *SAC 2013*, volume 8282 of *LNCS*. Springer, Heidelberg, August 2014.
→ Cited on pages 168, 177, and 178.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In *APPROX-RANDOM*, pages 450–461, 2006.
→ Cited on page 42.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact Knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 144–155. Springer, Heidelberg, July 2006.
→ Cited on pages 17, 21, and 61.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 54–72. Springer, Heidelberg, February 2008.
→ Cited on pages 61 and 113.

[LMW09] Felix Leder, Peter Martini, and André Wichmann. Finding and extracting crypto routines from malware. In *28th International Performance Computing and Communications Conference, IPCCC 2009, 14-16 December 2009, Phoenix, Arizona, USA*, pages 394–401, 2009.
→ Cited on page 160.

[LP15] Vadim Lyubashevsky and Thomas Prest. Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and gaussian sampling in structured lattices. In Oswald and Fischlin [OF15], pages 789–815.
→ Cited on pages 64, 80, 87, and 116.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23.
→ Cited on pages 17, 21, 47, 86, 101, and 113.

[LPR13a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. Preliminary version appeared in EUROCRYPT 2010.
→ Cited on pages 47, 61, 92, 137, 138, 139, 140, and 143.

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Johansson and Nguyen [JN13], pages 35–54.
→ Cited on pages 17, 21, 67, 86, 137, 138, and 139.

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 315–334. Springer, Heidelberg, August 2014.
→ Cited on page 47.

[LSL13] Laura Luzzi, Damien Stehlé, and Cong Ling. Decoding by embedding: Correct decoding radius and DMT optimality. *IEEE Transactions on Information Theory*, 59(5):2960–2973, 2013.
→ Cited on page 38.

[LSR+15] Zhe Liu, Hwajeong Seo, Sujoy Sinha Roy, Johann Großschädl, Howon Kim, and Ingrid Verbauwhede. Efficient ring-lwe encryption on 8-bit AVR processors. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 663–682, 2015.
→ Cited on page 159.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Nguyen and Oswald [NO14], pages 239–256.
→ Cited on page 47.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.
→ Cited on pages 76, 138, and 143.

[LvdPdW12] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2012/533, 2012. http://eprint.iacr.org/2012/533.
→ Cited on page 165.

[LW11] Dong Hoon Lee and Xiaoyun Wang, editors. *ASIACRYPT 2011*, volume 7073 of *LNCS*. Springer, Heidelberg, December 2011.
→ Cited on pages 167 and 170.

[Lyu08] Vadim Lyubashevsky. *Towards Practical Lattice-Based Cryptography*. PhD thesis, University of California, San Diego, 2008.
→ Cited on page 51.

[Lyu12a] Vadim Lyubashevsky. Lattice-based encryption. http://www.di.ens.fr/~lyubash/talks/LWEcrypto.pdf, 2012.
→ Cited on page 101.

[Lyu12b] Vadim Lyubashevsky. Lattice signatures without trapdoors. In Pointcheval and Johansson [PJ12], pages 738–755.
→ Cited on pages 101 and 137.

[MB64] G. Marsaglia and T. A. Bray. A convenient method for generating normal variables. *SIAM Review*, 6(3):pp. 260–264, 1964.
→ Cited on page 57.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.
→ Cited on pages 38 and 40.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, Heidelberg, August 1986.
→ Cited on page 16.

[Mor12] John Morris. Cracking the encrypted c&c protocol of the zeroaccess botnet. http://www.virusbtn.com/pdf/conference_slides/2012/Morris-VB2012.pdf, August 2012.
→ Cited on page 161.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [PJ12], pages 700–718.
→ Cited on pages 18, 86, 99, 112, 113, 116, 137, and 139.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Canetti and Garay [CG13], pages 21–39.
→ Cited on page 42.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.
→ Cited on pages 21 and 50.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.
→ Cited on pages 32, 51, 99, and 114.

[MT00] George Marsaglia and Wai W. Tsang. The Ziggurat Method for Generating Random Variables. *Journal of Statistical Software*, 5(8), 2000.
→ Cited on page 57.

[NO14] Phong Q. Nguyen and Elisabeth Oswald, editors. *EUROCRYPT 2014*, volume 8441 of *LNCS*. Springer, Heidelberg, May 2014.
→ Cited on pages 169 and 175.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Vaudenay [Vau06], pages 271–288.
→ Cited on pages 30, 34, 101, 143, and 164.

[NSA15] NSA. NSA suite B cryptography. https://www.nsa.gov/ia/programs/suiteb_cryptography/, 2015.
→ Cited on pages 21 and 165.

[Nus12] Henri J Nussbaumer. *Fast Fourier transform and convolution algorithms*, volume 2. 2012.
→ Cited on pages 11 and 113.

[NV10] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. 2010.
→ Cited on pages 30 and 179.

[OAG+11] Leonardo B. Oliveira, Diego F. Aranha, Conrado Porto Lopes Gouvêa, Michael Scott, Danilo F. Câmara, Julio López, and Ricardo Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011.
→ Cited on page 158.

[OAM+07] Leonardo B. Oliveira, Diego F. Aranha, Eduardo Morais, Felipe Daguano, Julio López, and Ricardo Dahab. Tinytate: Computing the tate pairing in resource-constrained sensor nodes. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007), 12 - 14 July 2007, Cambridge, MA, USA*, pages 318–323. IEEE Computer Society, 2007.
→ Cited on page 158.

[ODL+07] L.B. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A.A.F. Loureiro. Identity-based encryption for sensor networks. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 290–294, March 2007.
→ Cited on page 158.

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*. Springer, Heidelberg, April 2015.
→ Cited on pages 170 and 175.

[PDG14] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In Batina and Robshaw [BR14], pages 353–370.
→ Cited on pages 47, 49, 50, 137, and 164.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Rabin [Rab10], pages 80–97.
→ Cited on pages 18, 33, 35, 36, 50, 57, 85, 86, 90, 92, 113, 116, and 139.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219, 2014.
→ Cited on pages 156 and 164.

[PG12] Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In Hevia and Neven [HN12], pages 139–158.
→ Cited on page 164.

[PG14a] Thomas Pöppelmann and Tim Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *IEEE International Symposium on Circuits and Systemss, ISCAS 2014, Melbourne, Victoria, Australia, June 1-5, 2014*, pages 2796–2799. IEEE, 2014.
→ Cited on pages 159 and 164.

[PG14b] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In Lange et al. [LLL14], pages 68–85.
→ Cited on pages 138 and 164.

[PJ12] David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012.
→ Cited on page 176.

[POG15] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. High-performance ideal lattice-based cryptography on 8-bit atxmega microcontrollers. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, volume 9230 of *Lecture Notes in Computer Science*, pages 346–365, 2015.
→ Cited on pages 159 and 164.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 145–166. Springer, Heidelberg, March 2006.
→ Cited on pages 17, 21, and 61.

[PS07] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. Cryptology ePrint Archive, Report 2007/453, 2007. http://eprint.iacr.org/2007/453.
→ Cited on pages 156 and 162.

[PS09] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.
→ Cited on page 156.

[PS12a] Harsh Kupwade Patil and Stephen A. Szygenda. *Security for Wireless Sensor Networks Using Identity-Based Cryptography*. Auerbach Publications, Boston, MA, USA, 1st edition, 2012.
→ Cited on page 158.

[PS12b] Emmanuel Prouff and Patrick Schaumont, editors. *CHES 2012*, volume 7428 of *LNCS*. Springer, Heidelberg, September 2012.
→ Cited on page 172.

[PSY] P. Porras, H. Sadi, and V. Yegneswaran. A Multi-perspective Analysis of the Storm (Peacomm) Worm.
→ Cited on page 161.

[Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.
→ Cited on page 16.

[Rab10] Tal Rabin, editor. *CRYPTO 2010*, volume 6223 of *LNCS*. Springer, Heidelberg, August 2010.
→ Cited on pages 167 and 177.

[RD13] Christian Rossow and Christian J. Dietrich. Provex: Detecting botnets with encrypted command and control channels. In Konrad Rieck, Patrick Stewin, and Jean-Pierre Seifert, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment - 10th International Conference, DIMVA 2013, Berlin, Germany, July 18-19, 2013. Proceedings*, volume 7967 of *Lecture Notes in Computer Science*, pages 21–40, 2013.
→ Cited on page 161.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
→ Cited on pages 17, 21, 39, 42, 101, and 116.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
→ Cited on page 42.

[RG14] Steven Rich and Barton Gellman. NSA seeks to build quantum computer that could crack most types of encryption. https://www.washingtonpost.com/world/national-security/nsa-seeks-to-build-quantum-computer-that-could-crack-most-types-of-encryption/2014/01/02/8fff297e-7195-11e3-8def-a33011492df2_story.html, 2014.
→ Cited on page 165.

[RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. http://eprint.iacr.org/2010/137.
→ Cited on page 137.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.
→ Cited on pages 16 and 161.

[RVM+14] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact ring-LWE cryptoprocessor. In Batina and Robshaw [BR14], pages 371–391.
→ Cited on pages 159 and 164.

[RVV14] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High precision discrete gaussian sampling on FPGAs. In Lange et al. [LLL14], pages 383–401.
→ Cited on pages 18, 57, and 116.

[RZMT06] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In Jussara M. Almeida, Virgílio A. F. Almeida, and Paul Barford, editors, *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, pages 41–52. ACM, 2006.
→ Cited on page 159.

[SC09] Piotr Szczechowiak and Martin Collier. Tinyibe: Identity-based encryption for heterogeneous sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 319–354. IEEE, 2009.
→ Cited on page 158.

[Sch96] James C. Schatzman. Accuracy of the discrete fourier transform and the fast fourier transform. *SIAM J. Scientific Computing*, 17(5):1150–1166, 1996.
→ Cited on page 116.

[Sch14] Shane Schick. Ophionlocker ransomware uses advanced encryption to hold data hostage, 2014. https://securityintelligence.com/news/ophionlocker-ransomware-uses-advanced-encryption-to-hold-data-hostage/.
→ Cited on page 161.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.
→ Cited on page 38.

[Sec] Voltage Security. The identity-based encryption advantage - a proven standard for protecting information. https://www.voltage.com/resource/the-identity-based-encryption-advantage-a-proven-standard-for-protecting-information.
→ Cited on page 159.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In Blakley and Chaum [BC84], pages 47–53.
→ Cited on page 135.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.
→ Cited on page 21.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.
→ Cited on pages 135, 156, 158, and 162.

[Spi04] Terence Spies. Identity based encryption. http://csrc.nist.gov/archive/pki-twg/y2004/Presentations/twg-04-09.pdf, 2004.
→ Cited on page 159.

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47. Springer, Heidelberg, May 2011.
→ Cited on pages 99, 101, 138, 139, 143, and 146.

[SSPS13] Sérgio S. C. Silva, Rodrigo M. P. Silva, Raquel C. G. Pinto, and Ronaldo M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013.
→ Cited on page 160.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.
→ Cited on pages 17, 21, and 61.

[Ste98] G. W. Stewart. *Matrix Algorithms: Volume 1, Basic Decompositions*. Society for Industrial Mathematics, 1998.
→ Cited on page 65.

[Ste10] Damien Stehlé. Floating-point LLL: theoretical and practical aspects. In Nguyen and Vallée [NV10], pages 179–213.
→ Cited on page 65.

[Ste15] Noah Stephens-Davidowitz. Discrete gaussian sampling reduces to CVP and SVP. *CoRR*, abs/1506.07490, 2015.
→ Cited on pages 17 and 42.

[Swe84] D.R. Sweet. Fast toeplitz orthogonalization. *Numerische Mathematik*, 43:1–21, 1984.
→ Cited on page 63.

[Tsy08] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. 1st edition, 2008.
→ Cited on page 47.

[Vau06] Serge Vaudenay, editor. *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer, Heidelberg, May / June 2006.
→ Cited on pages 172 and 176.

[Ver10] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.
→ Cited on page 114.

[Wan10] Andrew Wan. *Learning, Cryptography, and the Average Case*. Phd thesis, Columbia University, 2010.
→ Cited on pages 30 and 34.

[WN95] David J. Wheeler and Roger M. Needham. TEA, a tiny encryption algorithm. In Bart Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 363–366. Springer, Heidelberg, December 1995.
→ Cited on page 160.

[WS12] Xiaoyun Wang and Kazue Sako, editors. *ASIACRYPT 2012*, volume 7658 of *LNCS*. Springer, Heidelberg, December 2012.
→ Cited on page 171.

[XWD10] Xiaokang Xiong, Duncan S. Wong, and Xiaotie Deng. Tinypairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6, April 2010.
→ Cited on page 158.

[YKH+06] Peng Yang, Takashi Kitagawa, Goichiro Hanaoka, Rui Zhang, Kanta Matsuura, and Hideki Imai. Applying fujisaki-okamoto to identity-based encryption. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 16th International Symposium, AAECC-16, Las Vegas, NV, USA, February 20-24, 2006, Proceedings*, pages 183–192, 2006.
→ Cited on page 147.

[YRV+06] Geng YANG, Chun-ming RONG, Christian VEIGNER, Jiang tao WANG, and Hong bing CHENG. Identity-based key agreement and encryption for wireless sensor networks. *The Journal of China Universities of Posts and Telecommunications*, 13(4):54 – 60, 2006.
→ Cited on page 158.

[Yu08] Ping Yu. *Direct Online/Offline Digital Signature Schemes*. PhD thesis, Denton, TX, USA, 2008. AAI3376067.
→ Cited on page 36.

[YY96] Adam L. Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *1996 IEEE Symposium on Security and Privacy*, pages 129–140. IEEE Computer Society Press, 1996.
→ Cited on page 160.

[Zak10] V. Zakorzhevsky. A new version of sality at large, 2010. http://www.securelist.com/en/blog/180/A_new_version_of_Sality_at_large.
→ Cited on page 160.

Abstract

Although rather recent, lattice-based cryptography has stood out on numerous points, be it by the variety of constructions that it allows, by its expected resistance to quantum computers, of by its efficiency when instantiated on some classes of lattices.

One of the most powerful tools of lattice-based cryptography is *Gaussian sampling*. At a high level, it allows to prove the knowledge of a particular lattice basis without disclosing any information about this basis. It allows to realize a wide array of cryptosystems. Somewhat surprisingly, few practical instantiations of such schemes are realized, and the algorithms which perform Gaussian sampling are seldom studied.

The goal of this thesis is to fill the gap between the theory and practice of Gaussian sampling. First, we study and improve the existing algorithms, by both a statistical analysis and a geometrical approach. We then exploit the structures underlying many classes of lattices and apply the ideas of the fast Fourier transform to a Gaussian sampler, allowing us to reach a quasilinear complexity instead of quadratic.

Finally, we use Gaussian sampling in practice to instantiate a signature scheme and an identity-based encryption scheme. The first one yields signatures that are the most compact currently obtained in lattice-based cryptography, and the second one allows encryption and decryption that are about one thousand times faster than those obtained with a pairing-based counterpart on elliptic curves.

Résumé

Bien que relativement récente, la cryptographie à base de réseaux euclidiens s'est distinguée sur de nombreux points, que ce soit par la richesse des constructions qu'elle permet, par sa résistance supposée à l'avènement des ordinateurs quantiques ou par la rapidité dont elle fait preuve lorsqu'instanciée sur certaines classes de réseaux.

Un des outils les plus puissants de la cryptographie sur les réseaux est le *Gaussian sampling*. À très haut niveau, il permet de prouver qu'on connaît une base particulière d'un réseau, et ce sans dévoiler la moindre information sur cette base. Il permet de réaliser une grande variété de cryptosystèmes. De manière quelque peu surprenante, on dispose de peu d'instanciations pratiques de ces schémas cryptographiques, et les algorithmes permettant d'effectuer du Gaussian sampling sont peu étudiés.

Le but de cette thèse est de combler le fossé qui existe entre la théorie et la pratique du Gaussian sampling. Dans un premier temps, nous étudions et améliorons les algorithmes existants, à la fois par une analyse statistique et une approche géométrique. Puis nous exploitons les structures sous-tendant de nombreuses classes de réseaux, ce qui nous permet d'appliquer à un algorithme de Gaussian sampling les idées de la transformée de Fourier rapide, passant ainsi d'une complexité quadratique à quasilinéaire.

Enfin, nous utilisons le Gaussian sampling en pratique et instancions un schéma de signature et un schéma de chiffrement basé sur l'identité. Le premier fournit des signatures qui sont les plus compactes obtenues avec les réseaux à l'heure actuelle, et le deuxième permet de chiffrer et de déchiffrer à une vitesse près de mille fois supérieure à celle obtenue en utilisant un schéma à base de couplages sur les courbes elliptiques.