PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée et soutenue à l'École normale supérieure de Paris

Extended Security of Lattice-Based Cryptography

Soutenue par
Mélissa Rossi
Le 10 septembre 2020

Ecole doctorale n° 386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Martin Albrecht Royal Holloway, University of London	*Examinateur*
Pierre-Alain Fouque Université de Rennes 1	*Président du jury*
Louis Goubin Université de Versailles-Saint-Quentin-en-Yvelines	*Rapporteur*
Helena Handschuh Rambus, San Francisco	*Examinatrice*
Daniele Micciancio University of California, San Diego	*Examinateur*
Damien Stehlé École normale supérieure de Lyon	*Rapporteur*
Michel F. Abdalla École normale supérieure de Paris	*Directeur de thèse*
Henri Gilbert ANSSI, Paris	*Codirecteur de thèse*

Invité·e·s ayant participé à l'encadrement :

Sonia Belaïd
CryptoExperts, Paris

Ange Martinelli
Thales, ANSSI, Paris

Thomas Prest
PQShield, Oxford

Guénaël Renault
École polytechnique, Palaiseau

ENS

THALES

Abstract

Lattice-based cryptography is considered as a quantum-safe alternative for the replacement of currently deployed schemes based on RSA and discrete logarithm on prime fields or elliptic curves. It offers strong theoretical security guarantees, a large array of achievable primitives, and a competitive level of efficiency. Nowadays, in the context of the NIST post-quantum standardization process, future standards may ultimately be chosen and several new lattice-based schemes are high-profile candidates. The cryptographic research has been encouraged to analyze lattice-based cryptosystems, with a particular focus on practical aspects. This thesis is rooted in this effort.

In addition to black-box cryptanalysis with classical computing resources, we investigate the extended security of these new lattice-based cryptosystems, employing a broad spectrum of attack models, e.g. quantum, misuse, timing or physical attacks. Accounting that these models have already been applied to a large variety of pre-quantum asymmetric and symmetric schemes before, we concentrate our efforts on leveraging and addressing the new features introduced by lattice structures. Our contribution is twofold: defensive, i.e. countermeasures for implementations of lattice-based schemes and offensive, i.e. cryptanalysis.

On the defensive side, in view of the numerous recent timing and physical attacks, we wear our designer's hat and investigate algorithmic protections. We introduce some new algorithmic and mathematical tools to construct provable algorithmic countermeasures in order to systematically prevent all timing and physical attacks. We thus participate in the actual provable protection of the GLP, BLISS, qTesla and Falcon lattice-based signatures schemes.

On the offensive side, we estimate the applicability and complexity of novel attacks leveraging the lack of perfect correctness introduced in certain lattice-based encryption schemes to improve their performance. We show that such a compromise may enable decryption failures attacks in a misuse or quantum model. We finally introduce an algorithmic cryptanalysis tool that assesses the security of the mathematical problem underlying lattice-based schemes when partial knowledge of the secret is available. The usefulness of this new framework is demonstrated with the improvement and automation of several known classical, decryption-failure, and side-channel attacks.

Résumé

La cryptographie fondée sur les réseaux euclidiens représente une alternative prometteuse à la cryptographie asymétrique utilisée actuellement, en raison de sa résistance présumée à un ordinateur quantique universel. Cette nouvelle famille de schémas asymétriques dispose de plusieurs atouts parmi lesquels de fortes garanties théoriques de sécurité, un large choix de primitives et, pour certains de ses représentants, des performances comparables aux standards actuels. Une campagne de standardisation post-quantique organisée par le NIST est en cours et plusieurs schémas utilisant des réseaux euclidiens font partie des favoris. La communauté scientifique a été encouragée à les analyser car ils pourraient à l'avenir être implantés dans tous nos systèmes. L'objectif de cette thèse est de contribuer à cet effort.

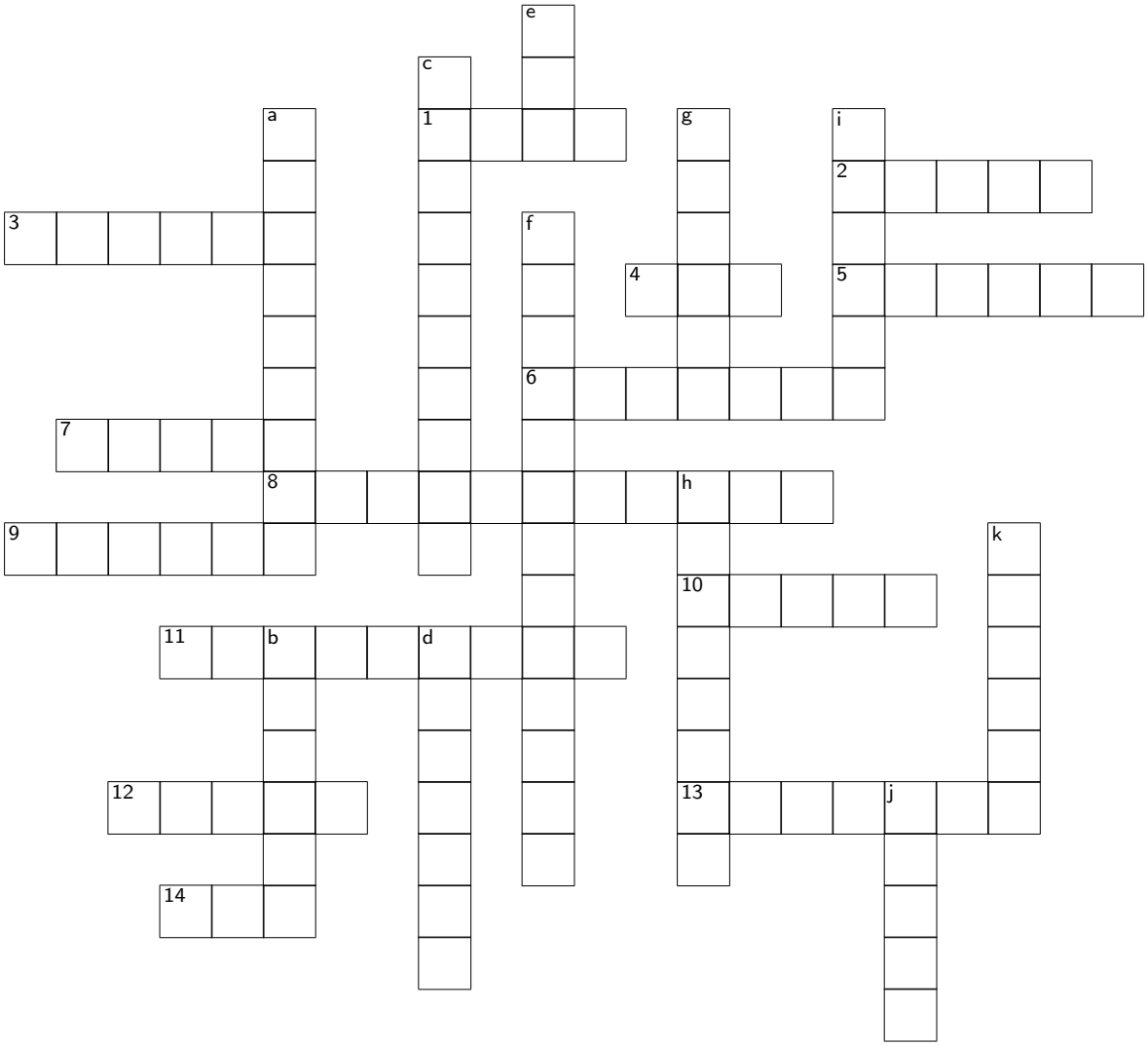
Nous étudions la sécurité de ces nouveaux cryptosystèmes non seulement au sens de leur résistance à la cryptanalyse en "boîte noire" à l'aide de moyens de calcul classiques, mais aussi selon un spectre plus large de modèles de sécurité, comme les attaques quantiques, les attaques supposant des failles d'utilisation, ou encore les attaques par canaux auxiliaires. Ces différents types d'attaques ont déjà été largement formalisés et étudiés par le passé pour des schémas asymétriques et symétriques pré-quantiques. Dans ce mémoire, nous analysons leur application aux nouvelles structures induites par les réseaux euclidiens. Notre travail est divisé en deux parties complémentaires : les contremesures et les attaques.

La première partie regroupe nos contributions à l'effort actuel de conception de nouvelles protections algorithmiques afin de répondre aux nombreuses publications récentes d'attaques par canaux auxiliaires. Les travaux réalisés en équipe auxquels nous avons pris part on abouti à l'introduction de nouveaux outils mathématiques pour construire des contre-mesures algorithmiques, appuyées sur des preuves formelles, qui permettent de prévenir systématiquement les attaques physiques et par analyse de temps d'exécution. Nous avons ainsi participé à la protection de plusieurs schémas de signature fondés sur les réseaux euclidiens comme GLP, BLISS, qTesla ou encore Falcon.

Dans une seconde partie consacrée à la cryptanalyse, nous étudions dans un premier temps de nouvelles attaques qui tirent parti du fait que certains schémas de chiffrement à clé publique ou d'établissement de clé peuvent échouer avec une faible probabilité. Ces échecs sont effectivement faiblement corrélés au secret. Notre travail a permis d'exhiber des attaques dites « par échec de déchiffrement » dans des modèles de failles d'utilisation ou des modèles quantiques. Nous avons d'autre part introduit un outil algorithmique de cryptanalyse permettant d'estimer la sécurité du problème mathématique sous-jacent lorsqu'une information partielle sur le secret est donnée. Cet outil s'est avéré utile pour automatiser et améliorer plusieurs attaques connues comme des attaques par échec de déchiffrement, des attaques classiques ou encore des attaques par canaux auxiliaires.

Remerciements

En arrivant de San Francisco, je ne pensais pas que cette expérience serait aussi riche en émotions. Je m'imaginais encore moins qu'elle se terminerait par une pandémie mondiale et une soutenance en comité restreint. Mais cela n'enlève rien à mon enthousiasme pour ce domaine passionnant et ma fierté au regard du chemin parcouru. Voici quelques mots-clés qui, à mon avis, ont participé au bon déroulement et à l'authenticité de cette thèse.



Lignes

1 Super allié et boosteur de moral. Je ne peux pas en dire plus, par crainte de la mafia corse.

2 Activité de détente fréquemment pratiquée avec Louiza et Chloé qui consistait à se mettre de la terre dans les cheveux pour devenir plus ou moins rousse.

3 Quizz quotidien de culture générale dont les réponses sont principalement détenues par Jérôme ou Hugues.

4 Nombre anormalement élevé d'affiliations pour une doctorante. Je ne regrette pas ce format original de thèse multipartite qui m'a permis d'être entourée de collègues brillants issus de milieux académiques, industriels et étatiques.

5 "Parce que tu le vaux bien", et parce que le prix Jeune Talents a été une incroyable expérience. Je remercie toute son équipe chaleureusement.

6 Passion très contagieuse partagée avec plusieurs membres du laboratoire de l'ENS. Victimes de selection naturelle pendant le confinement.

7 Dira qu'il n'a pas assez dirigé ma thèse alors que son soutien a été sans faille quelle que soit l'heure du jour et de la nuit. Merci infiniment !

8 Groupe de recherche presque totalement féminin relié au projet RISQ.

9 Ornithophile arachnophobe toujours disponible pour m'aider malgré son emploi du temps chargé, source perpétuelle des bonnes idées de recherche.

10 A l'origine de mon projet de thèse, elle m'a guidée et conseillée avec beaucoup de bienveillance. J'ai beaucoup apprécié travailler avec elle, merci !

11 Mon premier jour de thèse a consité à distribuer des badges et goodies pendant cette conférence.

12 Quoi qu'en disent les rageux, c'est définitivement un sport.

13 Japonophile qui m'a aidée à comprendre les algorithmes de bases de Gröbner. Merci pour ton soutien et tes conseils pendant ma thèse !

14 Structure académique prestigieuse dans laquelle j'ai été ravie de pouvoir faire ma thèse. Je n'oublierai pas les nombreuses pauses café à refaire le monde avec les membres de l'équipe.

Colonnes

a Un des nombreux acronymes douteux générés pendant cette thèse.

b *Great Californian company where I had the chance to make a pre-doc internship.*

c Contrairement au cryptnic, je n'ai jamais pu en organiser par manque de volontaires.

d La raison pour laquelle il faut se faire discret en automne et au printemps.

e Seul projet de recherche qui n'avait rien à voir avec mon pourtant très large sujet de thèse, mais qui m'a permis de travailler avec des amis dans une super ambiance.

f Ce n'est pas une startup mais une PME : environnement très stimulant avec une équipe de choc que j'ai été ravie de venir voir régulièrement.

g Boss final à convaincre pour pouvoir soutenir. Ses remarques ont grandement amélioré la qualité de ce manuscrit.

h Principe du chapitre 3 et aussi une des solutions à l'épidémie de Covid19.

i Environnement industriel dynamique aux mille avantages dans lequel j'ai passé de très belles années. Conditions idéales pour un thèse Cifre !

j Je suis ravie d'avoir atterri dans son équipe de crypto et très enthousiaste à l'idée d'y passer les prochaines années.

k En plus de ses fonctions présidentielles, il a été un guide et conseiller expérimenté tout au long de ma thèse.

J'aimerais exprimer plus en détails ma gratitude à toutes les personnes qui ont contribué à faire de ma thèse l'aventure la plus épanouissante que j'ai connue. Mes pensées se dirigent d'abord vers Henri Gilbert, mon mentor et directeur de thèse de l'ANSSI. Je le remercie pour ses idées précieuses, ses enseignements, ses encouragements et son manque d'objectivité à mon égard. Je remercie aussi Michel F. Abdalla, mon directeur de thèse de l'ENS, pour ses conseils et sa bienveillance.

Je tiens à exprimer mes sincères remerciements à Damien Stehlé et Louis Goubin pour avoir accepté d'être rapporteurs de mon manuscrit de thèse. *I sincerely thank Martin Albrecht, Pierre-Alain Fouque, Helena Handschuh, and Daniele Micciancio for agreeing to be part of my jury.* Mes remerciements se dirigent ensuite vers Henri, Ange, Thomas Prest et Ricosset, Jean-René, Sonia, Hugues et Natacha Laniado pour leurs relectures précieuses de ce manuscrit.

Je tiens à remercier Didier Le Maitre et Éric Garrido qui m'ont accueillie au sein du laboratoire de cryptographie de Thales SIX pour ma thèse. C'était un environnement parfait pour bénéficier d'un point de vue industriel tout en ayant de la liberté dans mes recherches. J'y ai semble-t-il installé un siège éjectable pour mes encadrant·e·s de thèse. Je remercie d'abord chaleureusement Sonia Belaïd qui a été à l'origine de ce projet de thèse ambitieux et qui m'a accompagnée du début à la fin. Son départ de Thales n'a rien retiré à notre collaboration et j'en suis très heureuse. J'ai vraiment apprécié notre travail ensemble, autant l'ambiance bienveillante que nos réalisations. Merci ensuite à Thomas Prest pour s'être assis en second sur ce siège éjectable. Tu m'as transmis ton instinct paternel pour un certain oiseau et tu as fait croître mon interêt pour la Rényi divergence. Ton éjection à Oxford ne nous a pas non plus empêchés de collaborer et je t'en remercie. Merci enfin à Ange Martinelli pour s'être assis sur ce siège éjectable en dernier. Ton soutien indéfectible et ton goût en jeux de société m'ont été d'une grande aide. Je regrette que nous n'ayons pas eu le temps de collaborer sur un papier mais ce n'est que partie remise puisque, par un heureux hasard, ton siège éjectable t'a envoyé à quelques pas de mon bureau actuel. Désolée Éric pour leur départ à tous ! Je remercie aussi toute l'équipe du laboratoire de Thales: Thomas Ricosset, fidèle partenaire de soirées jeux de sociétés, Aurélien Dupin, partenaire de voyages, Renaud Dubois, David Lefranc, Olivier Bernard, Philippe Painchault, Olivier Orcière, Matthieu Giraud, Julien Prat, Sylvain Lachartre, Zoé Amblard que j'ai initiée à la Zumba, Anne Viguie et Simon Masson, mon 'co-couloir' aux super talents de magicien. Je remercie aussi les anciens du laboratoire: Alexandre Yamajako-Anzala, Émeline Hufschmitt et Michael Geffrault. Merci à Eric Sageloli d'avoir fait un stage aussi brillant avec nous ! Je remercie aussi les collègues et ami·e·s de l'étage du dessus : Nicolas Waroquier, Maxime Peycelon et Marion Beurard. J'ai passé de superbes années dans cette structure dynamique. Qu'aurait été ma thèse sans le club ZumbaWarrior, l'équipe des cryptosaures et les voyages du CE ?

J'ai eu la chance d'atterrir dans le laboratoire de cryptographie de l'ENS de Paris pour l'encadrement académique. J'aimerais adresser mes remerciements à David Pointcheval qui m'a accueillie dans son équipe. En trois ans et demi, j'ai pu y passer de mémorables moments et y créer de belles amitiés. Merci particulièrement à mes ami·e·s et partenaires de voyages Chloé Hébant, toujours partante pour les activités extravagantes (brunch-tricot, henné-raclette...), mon amie plantophile Louiza Khati, Aurélien Dupin et ses super suggestions de jeux de société, Antoine Plouviez, l'organisateur du jeu (prémonitoire ?) de Spors et dessinateur émérite, Romain Gay et ses innombrables mal-chances, Pierrick Méaux et sa passion pour les dahus, Quoc Huy Vu toujours partant pour les activités avec le labo et Anca Nitulescu dont la taille de la garde-robe parait infinie. Merci aussi aux autres membres de l'équipe : Brice Minaud, David Naccache, Hoeteck Wee, Céline Chevalier, Phong Nguyen, Duong-Hieu Phan, Azam Soleimanian, Balthazar Bauer, Léo Colisson, Luka Music, Théo Ryffel, ainsi qu'aux nouveaux Baptiste Cottier, Lénaick Gouriou et Hugo Senet. Merci aussi aux anciens du laboratoire : Georg Fuchsbauer, Damien Vergnaud, Ehsan Ebrahimi, Pooya Farshim, Junqing Gong, Julia Hesse, Fabrice

Benhamouda, Florian Bourse, Jérémy Chotard, Aisling Connolly, Geoffroy Couteau, Rafaël Del Pino, Pierre-Alain Dupont, Dahmun Goudarzi, Thierry Mefenza, Michele Minelli, Michele Orrù, Alain Passelègue, Razvan Rosie, Léonard Assouline, Thibaut Bagory, Hugo Marival, Edouard Dufour Sans, Bogdan Ursu et Marie Euler. Merci aussi aux personnels administratifs qui m'ont aidée et suivie : Stéphane Émery, Lise-Marie Bivard, Sophie Jaudon, Valérie Mongiat et Jacques Beigbeder.

J'ai aussi eu l'occasion d'arpenter régulièrement les couloirs de l'ANSSI durant ma thèse. J'y ai rencontré de très bons collègues et co-auteurs dans une ambiance bienveillante ponctuée de parties de Quidol et de soirées Miyazaki. J'aimerais adresser mes remerciements à toute l'équipe et notament Guénaël Renault qui a aussi participé à mon encadrement côté ANSSI. Merci à Aurélie Bauer, inspirante par ses qualités de pédagogie, Jérôme Plût et toutes ses inventions, Jérémy Jean, Jean-René Reinhard, Hugues Randriam, Yannick Seurin, Thomas Fuhr et aussi Colin ~~Chaigneau~~ Chataigneau pour ses cookies. Je n'oublie pas non plus Thomas Trouchkine, David El-baze, Guillaume Bouffard, Emmanuel Prouff, Adrian Tillard, Ryad Benadjila, Karim Khalfallah, Boris Simunovic, Yoan Chergui et Eliane Jaulmes. J'ai finalement été invitée à poser mes valises à l'ANSSI après ma thèse et je suis très honorée et heureuse de faire partie de l'équipe. Merci beaucoup à Henri, José Araujo, Laura Coman et Isabelle Pastor d'avoir rendu mon arrivée possible. Merci à Thibauld Feneuil d'avoir fait un remarquable stage à l'ANSSI et CryptoExperts, en plein confinement.

J'aimerais bien sûr également remercier chaleureusement tous mes co-auteurs. Ces derniers ont été des sources d'inspiration et d'admiration durant toute ma thèse. Dans l'ordre chronologique, je voudrais remercier l'équipe latmasking composée de Thomas Espitau, Mehdi Tibouchi qui a plusieurs fois été mon joker pour les questions difficiles, Pierre-Alain Fouque, Gilles Barthe, Sonia Belaïd et Benjamin Grégoire. Ensuite, je remercie Aurélie Bauer, Guénaël Renault et Henri Gilbert, mes co-auteurs de cryptanalyse à l'ANSSI. Merci aussi à l'équipe de choc pour la 'PRG-party', Aurélien Dupin, Pierrick Méaux, Geoffroy Couteau et Yann Rotella. Je remercie François Gérard pour ses idées pertinentes et ses talents d'implémentation. Merci à Thomas Prest et Ricosset et James Howe pour notre collaboration pour rendre Falcon toujours plus compétitif. *I would like to thank the LeakLWE team for getting the opportunity to work on this awesome project: Léo Ducas, Huijing Gong and Dana Dachman-Soled. I also thank the failmore team with whom I had the pleasure to work with, Jan-Pieter D'Anvers, Fernando Virdia, Alessandro Budroni and Henri who helped us a lot. Besides, I would also like to thank all the anonymous reviewers of the papers submitted during this thesis.* Merci aussi aux équipes dans lesquelles les projets n'ont pas pu aboutir, j'y ai appris beaucoup. Merci à Michelino Orrù et Michel F. Abdalla avec qui nous avons eu le faux-espoir de pouvoir attaquer et prouver une protection des signatures contre les RKA. Merci aussi à l'équipe cryptowomen comportant Malika Isabachène, Natasha Kharchenko, Sonia, Aurélie et Henri (cherchez l'erreur ;)), avec qui nous avons démêlé les sombres mécanismes des échecs de déchiffrement. Merci à Mehdi Tibouchi avec qui j'aurais aimé faire un projet à NTT mais c'était sans compter les méandres de l'administration japonaise.

J'aimerais aussi remercier plusieurs cryptologues rencontrés lors de conférences ou de collaborations pendant ces trois riches années. *I would like to thank all the Rambus team and particularly Mark E. Marson, Helena Handschuh, Mike Hamburg and Mike Hutter for a very prolific introduction to research in post-quantum cryptography.* Merci beaucoup à Sonia, Matthieu Rivain, Pascal Paillier et Louis Goubin pour m'avoir accueillie très chaleureusement à CryptoExperts. J'ai été ravie d'y rencontrer Natasha Kharchenko et Junwei Wang. Je remercie aussi Léo Ducas ainsi que toute l'équipe de CWI pour m'avoir accueillie à plusieurs occasions. Merci aussi à Alice Pellet-Mary, partenaire de soirées crèpes improvisées, à Yannick Sierra avec qui nous avons exploré Taïwan, à François-Xavier Standaert pour l'invitation à Louvain-La-Neuve. *I thank Paul Kocher for fruitful discussions during conferences. I would like to thank Gustavo Banegas for the interesting conversations on quantum gate counting and Tanja Lange for her invitation*

at a workshop on post-quantum cryptography. Thanks also to the organizers of CryptoActions workshop for their invitation. J'aimerais aussi remercier mes partenaires de promo du MPRI qui ont aussi choisi de faire de la crypto : Thomas Debris-Alazard, Thomas Espitau, Xavier Bonnetain, Marius Lombard-Platet et Luka Music (la crypto quantique ça compte !). Je souhaite aussi remercier deux professeurs de Télécom, Anne Canteaut et Hugues Randriam, désormais collègue de l'ANSSI, qui m'ont transmis leur passion de la cryptographie et sans qui je ne serais pas arrivée ici.

J'ai eu la chance de prendre part à plusieurs projets de vulgarisation pour inciter les jeunes adolescents, et surtout adolescentes, à faire de l'informatique. Merci à l'équipe de la Méthode Scientifique de France Culture pour m'avoir invitée à parler de ma thèse, à David Blottière pour m'avoir invitée dans sa classe prépa, à l'association Coding Sisters, à Chi Tran pour son interview Youtube, à Melvin Riquier pour la super vidéo animée et à Laure Delalex pour son projet de documentaire. Je suis aussi très reconnaissante au programme Google Women Tech Makers pour m'avoir décerné une bourse et permis de faire partie de leur réseau. Je remercie aussi chaleureusement toute l'équipe de la bourse L'Oréal UNESCO ainsi que les co-lauréates du prix Jeunes Talents 2019 pour cette superbe expérience.

Dans un esprit plus personnel, je remercie ma chorale pour les moments de détente musicale du mercredi soir. J'aimerais aussi remercier les amies qui m'ont soutenue et ont suivi mon projet de thèse depuis le début. Je remercie les Télécommédiennes Muriel, Chiara et Elena. Merci également à Marion Beurard, amie précieuse qui m'a initiée au féminisme. Je remercie aussi évidemment mes éternelles copines de l'Hawanawa : merci à Cindy chez qui je m'exile de temps en temps pour fuir le tumulte parisien, à Amandine et Alexandra.

J'adresse de profonds remerciements à ma famille qui a fait preuve d'un soutien indéfectible durant ces trois années, mais aussi durant toutes celles qui les ont précédées. Je remercie mes petites soeurs Olivia et Mathilde avec qui nous formons un trio complice, digne des Totally Spies. Je remercie également mon père, toujours disponible pour aller se changer les idées et profiter de la vie. Un grand merci à ma mère pour être la mère la plus formidable qui existe. J'espère que le doctorat ne mettra pas fin aux récompenses colliers-bonbons ! Je n'oublie évidemment pas Popi que je remercie pour toutes ses qualités canines, ainsi que Pelote pour son flegme. Je remercie également mes grands-parents, toujours là pour me chouchouter quand il le fallait. Je dédie cette thèse à ma grand-mère, qui l'a vécue avec moi et m'a toujours soutenue, en toute objectivité.

Enfin, j'aimerais remercier Romaric pour être un merveilleux allié et pour le bonheur qu'il me procure en partageant ma vie. Nous nous sommes entraidés dans les (nombreux) moments de stress pour nos thèses, moi sur les lattices en cryptographie et lui sur les lattices cristallins des matériaux semi-conducteurs.

Contents

Chapter

1 Introduction

To improve social distancing, all lattices must guarantee that they contain no non-zero vector shorter than 2 meters to be considered safe. SVP-COVID.

@FakeIACR

Chapter content

Cryptography, the science dedicated to protecting data and communications, has become prevalent within a few decades. It can now be found at an industrial scale in smart cards, mobile phones, all over the internet, etc. Cryptography can rely on a secret key, which is shared between the sender and the receiver and nobody else. Such protocols belong in the *symmetric cryptography* family, and require the sender and the receiver to agree on a common secret key, before being able to communicate safely. Deriving a common secret key can be achieved thanks to another category denoted *asymmetric cryptography*, also called *public-key cryptography*. The concept has been introduced by Diffie and Hellman in 1976 [DH76]. The principle is that each user now has a publicly available key, called *public key* related to its own private key. The latter enables anyone to encrypt a message using the public key and, only the decryption is only possible with the corresponding private key. In today's cryptographic protocols, for performance

reasons, a public-key scheme is typically used to derive a shared secret symmetric key between two users. Then, symmetric cryptography, usually more efficient, is typically used for the rest of the protocol. This thesis exclusively focuses on asymmetric cryptography.

The asymmetric cryptographic security guarantees are based on the conjectured hardness of mathematical problems. Hard problems – which concretely means that even putting together a lot of computing power will not allow solving them – can be used to build communication protocols that are guaranteed to be secure. Almost all implemented cryptographic protocols base their security on the hardness of two mathematical problems: the integer factorization and the discrete logarithm. Unfortunately, both problems are threatened by the potential advent of large scale quantum computers[1].

Quantum computers are a new type of computers whose instructions are not based on classical physical principles, but rather on quantum physics. The lowest-level brick of a quantum computer is a quantum bit or qubit. In contrast to a classical bit, a qubit can take infinitely many other values (called quantum superpositions) than 0 and 1. This feature allows quantum computers to be much faster than classical computers over some classes of problems, among which the factorization and discrete logarithm [Sho94]. As a result, all protocols based on these problems would be rendered insecure by a large-scale quantum computer. Progress in quantum computing is slow but steady: the current quantum computers can manipulate about a hundred physical qubits [AAB19] while more than several thousands logical[2] qubits are necessary for breaking the factorization. Some specialists expect that practical large scale universal[3] quantum computers might become a reality within the next decades [Che+16].

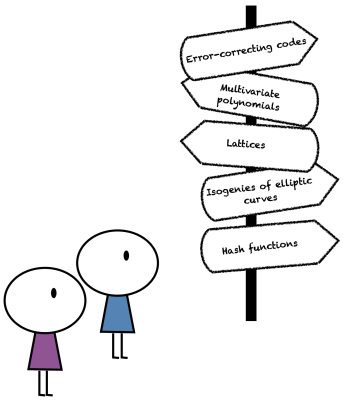
To prevent a collapse of cryptographic protocols, and to avoid a mono-culture of hardness assumptions, researchers have worked on building secure cryptographic protocols which do not rely on the factorization and discrete logarithm, and that can be expected to be impervious to attacks performed by quantum computers. These schemes are usually labeled under the umbrella terms "quantum-safe cryptography" or "post-quantum cryptography". Post-quantum cryptography has gained momentum these last few years, driven both by advances in quantum computing and by the fact that information that needs to be protected nowadays may still be sensitive by the time quantum computers become available. Indeed, long-term sensitive encrypted data can be stored today and they can potentially be decrypted later, if a quantum computer is built. Another illustration is the emerging standardization initiatives around the world. More precisely, the NIST (National Institute of Standards and Technology, USA) is currently running a standardization campaign for post-quantum protocols that started in fall 2017 [Che+16]. Future standards are expected in a few years.

Several promising pathways exist towards robust asymmetric post-quantum schemes, in particular those relying on the hardness of problems over lattices, error-correcting codes, multivariate polynomial systems, isogenies of elliptic curves or on the security of hash functions. All these problems are expected to remain hard even in the presence of a quantum computer. *Lattice-based cryptography* has been widely recognized as a foremost candidate for practical, post-quantum security and significant efforts have been devoted to develop and analyze lattice-based cryptosystems. In particular, it offers strong theoretical security guarantees, a large array of achievable primitives, and a level of efficiency that can compete, at least for the so-called structured instances, with pre-quantum constructions. This thesis is rooted in this context and focuses on analyzing lattice-based cryptosystems.

[1] For a 4-minutes introductory presentation the post-quantum cryptography, please refer to this France Culture Podcast recorded in French in the beginning of my PhD.

[2] There is a multiplicative factor accounting for error correcting codes necessary to convert physical qubits into logical qubits. This factor is currently estimated at around a thousand.

[3] The universality for a quantum computer means that its action is not only confined to a fixed circuit but it can be reprogrammable for executing any polynomial-time algorithm. Achieving the universality for a quantum computer is also challenge in quantum computing.



1.1 Cryptanalysis

The cryptanalysis is a subdomain of cryptology which consists in attacking existing cryptographic constructions in order to challenge their claimed security properties. Its purpose is to improve the cryptosystems' security by understanding their weaknesses. In the context of an attack against a given cryptosystem, an "attack model" should be defined containing two important parts:

1. The *target* of the attack must be clearly stated. Its purpose can be to recover a private key, the content of an encrypted message or to forge a signature for instance. In all this thesis, the focus is on recovering the private key on lattice-based cryptosystems.

2. The *attack resources* must be also specified. They represent the means that are available to the attacker. There exists a multitude of resources based on different use cases. The less means are available to the attacker, the more powerful the attack is. First, the computational resources available for the attacker should be quantified. These resources include time and memory. Second, the attack model should also specify what information is available to the attacker, the minimum being the public parameters (like the public key). As shown in the next subsections, depending on the context, some additional information and ressources may be available to the attacker.

1.1.1 Classical attacks

An attacker always knows at least the public parameters of a given cryptosystem and thus its public key. She may want to recover the associated private key. A classical cryptanalysis only uses this information to estimate the amount of computing ressources required to recover the private key. Sometimes, depending on the model, the attack can have access to decryption or signature oracles. The classical attacks often rely on breaking the underlying hard problem. Such classical attack models have been studied since the beginning of cryptography and it is now an important part of the design of new cryptosystems.

1.1.2 Quantum attacks

Quantum attacks started to be studied after Shor's quantum algorithm for period finding in 1994 [Sho94]. In quantum attacks, we assume that the attacker has access to *quantum algorithms*. The latter are faster for certain determined tasks and they are likely to be implemented in the future on large quantum circuits. Quantum computing falls outside the scope of this thesis but some quantum algorithms are used in attacks as a "black-box".

Grover's algorithm. We will use for example Grover's algorithm [Gro96] in Section 4.3. It is assumed to be able to find with high probability the unique input of a function that produces a particular output value using just $O(\sqrt{N})$ evaluations of the function, where N is the cardinal of the function's domain.

1.1.3 Side-channel attacks

Timing attacks. In timing attacks, we assume that the attacker is able to measure the time taken to execute cryptographic algorithms. Indeed, every algorithm in a computer takes time to execute, and the duration can differ based on the input and the private key. For example, the execution time of a square-and-multiply algorithm used in modular exponentiation depends linearly on the number of bits "1" in the private key. With several measurements and statistical techniques, an attacker is able to recover the private key. In the context of cryptanalysis, this model of attacks was formalized by Kocher in 1996 [Koc96]. In his work, he showed that these attacks were practical against a number of encryption algorithms, including RSA, ElGamal, and the Digital Signature Algorithm. Nowadays, timing attacks are an important part of cryptanalysis [Lip+18; BB03; Por18; Ber05]. Along with the physical attacks presented in the next subsection, they are in the side-channel attacks family.

Physical attacks. Physical attacks are a family of powerful attacks that target an algorithm practical implementation on physical chips. They were introduced and formalized in 1999 [KJJ99; GP99; Cha+99]. In this setting, the corresponding attackers can observe the device's emanations (electromagnetic waves, acoustic waves, temperature or power consumption) to recover the private key. These physical attacks can be categorized in many different ways, depending on the adversary's capabilities, the available equipments or their complexity. We refer to [MOP07] for a complete description.

We outline hereafter two types of non-invasive[4] physical attacks which can extract the secret key from the device. Both exploit *traces of leakage* of the device, which are the graphs representing a measured physical leakage (e.g. power consumption) during the execution of an algorithm as a function of the time. The physical leakage is generally correlated to changes in the state of the registers and then to the calculation processed. These two types of physical attacks are the following.

- A *single-trace analysis* consists in extracting information from the visual interpretation of one or very few traces. This type of attack often targets algorithms that are executed only once like key generations. We mention this type of attack in Section 5.6.1.

- In a *differential analysis* an attacker computes intermediate values computed in the algorithm and uses statistical analysis of multiple traces.

1.1.4 Misuse attacks

One may also assume that, for example for reasons related to some practical constraints, the implementation of a cryptosystem slightly differs from the theoretical specifications. In other words, some modifications like the reuse of randomness or private key may be implemented in order to increase the efficiency while outside from the recommandations. This scenario has been called *misuse* in [RS06]. We investigate a misuse case in Section 4.2.

[4] Some invasive attacks consists in injecting faults during the execution of an algorithm. They are not considered in this thesis.

1.2 Lattice-based cryptography

In this thesis, we denote by $\mathbb{Z}_q$ the ring $\mathbb{Z}/q\mathbb{Z}$. We write $\coloneqq$ for a deterministic assignation and $\leftarrow$ for a probabilistic assignation either from a probability distribution or from a function with probabilitic output. For S a finite set, we also note $\xleftarrow{\$} S$ for drawing uniformly at random an element in S. For $m \in \mathbb{N}$, we define the ℓ_2-norm on $\mathbb{R}^m$ as $\|\mathbf{x}\|_2 \coloneqq \sqrt{\sum_i \mathbf{x}_i^2}$. We use row notations for vectors, and start indexing from 0. The notation I_n denotes the identity matrix of dimension n.

1.2.1 Lattices

A lattice is essentially a set of points in the space arranged periodically. This structure has been commonly used in many disciplines, from art[5] to crystallography[6]. Fig. 1.1 presents a two-dimensional lattice. This structure has received considerable attention from mathematicians starting with the early works of Gauss around the beginning of the XIX century, Hermite and Minkowski around the end of the XIX century and Lenstra, Lenstra and Lovasz in the 80's to quote a few names. Lattices started to be used as a tool for analyzing the security of a variety of cryptographic problems. Later, they have been shown to possess some unique properties in computational complexity leading to the current cryptographic constructions studied nowadays.

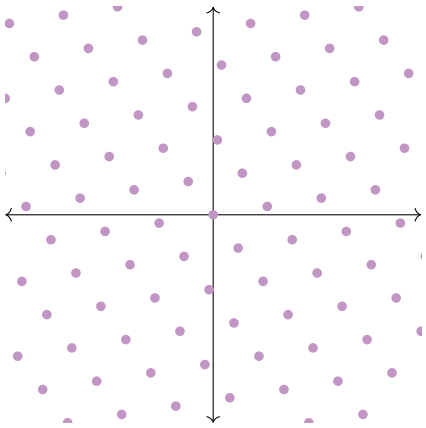


Figure 1.1: *A two-dimensional lattice*

A lattice is an infinite discrete structure generated by a finite set of vectors called *basis* $(\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$. Informally, any lattice is defined by a small unit parallelogram defined by these n vectors $(\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$ in $\mathbb{R}^m$.

More precisely, a lattice is generated as the set of all linear integer combinations of n $(n \leq m)$ linearly independent basis vectors $\{\mathbf{b}_j\} \subset \mathbb{R}^m$, namely,

$$\mathcal{L} \coloneqq \left\{ \sum\nolimits_j z_j \mathbf{b}_j : z_j \in \mathbb{Z} \right\}.$$

We say that m is the *dimension* of Λ and n is its rank. A lattice is *full rank* if $n = m$. The *volume* of a lattice is defined as $\mathrm{Vol}(\mathcal{L}) \coloneqq \sqrt{\det(\mathbf{B}\mathbf{B}^T)}$ where $\mathbf{B}$ is defined as a matrix in $\mathbb{R}^{n \times m}$ whose rows are the vectors of the basis.

For simplicity, we will depict lattices here in two dimensions but in cryptography, one must think of a high-dimensional structure.

[5] https://en.wikipedia.org/wiki/Latticework
[6] https://en.wikipedia.org/wiki/Bravais_lattice

1.2.2 Geometric lattice-based hard problems

As previously stated, public-key cryptography bases its security guarantees on the conjectured hardness of mathematical problems like the discrete logarithm or the factorization problem. Equivalently, several recently-defined hard problems are the building bricks for lattice-based cryptography. In this thesis, we formulate hard problems as challenges using the key words "given" and "find" for search problems or "given" and "distinguish" for decisional problems. The lattice-based hard problems have interesting, actively-studied properties (strong computational hardness, worst-case to average-case reductions) [LLL82; GG98; AR04; Kho04] that place lattice-based cryptography as a high-profile direction for post-quantum cryptography.

Let us first introduce a notation for the closed zero-centered Euclidean hyperball: for any $m \in \mathbb{N}$, $\mathbf{c} \in \mathbb{R}^m$ and $r > 0$,

$$B_m(\mathbf{c}, r) \coloneqq \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x} - \mathbf{c}\|_2 \leq r\}. \tag{1.1}$$

This ball is used to define the successive minima of a lattice. These are the sizes of linearly independent vectors in $\mathcal{L}$ ordered by increasing norm. Given a lattice $\mathcal{L}$, we denote by $\lambda_i(\mathcal{L})$ the i-th minimum of $\mathcal{L}$, as follows.

$$\lambda_i(\mathcal{L}) \coloneqq \inf \{r > 0 : \dim (\mathrm{Span} (\mathcal{L} \cap B_m(0, r))) \geq i\}.$$

For a better intuition, two of them are graphically represented in Fig. 1.2.

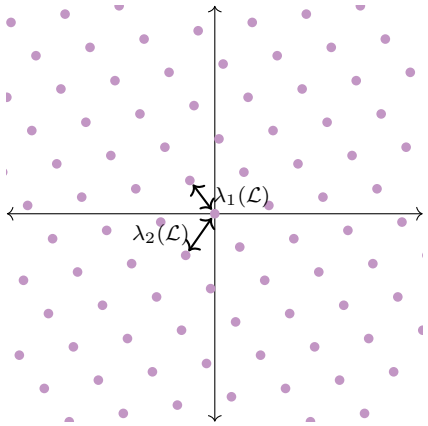


Figure 1.2: *Successive minima on a two-dimensional lattice*

Let us now introduce two important geometric problems on lattices.

Hard Problem 1 — Unique Shortest Vector Problem (uSVP)

Let n and m be positive integers and $\rho \geq 1$ a parameter.

Given a basis $(\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L} \subset \mathbb{R}^m$ that verifies $\lambda_2(\mathcal{L}) > \rho \cdot \lambda_1(\mathcal{L})$

Find a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\|_2 = \lambda_1(\mathcal{L})$

When $\rho = 1$, the problem is called *Shortest Vector Problem* (SVP).

We introduce another hard problem, that, instead of searching for the shortest vector, searches for the *closest* to a given point from $\mathbb{R}^m$. For any point $\mathbf{t} \in \mathbb{R}^m$ and any lattice $\mathcal{L}$, the distance of $\mathbf{t}$ to the lattice is denoted $dist(\mathbf{t}, \mathcal{L}) \coloneqq \min\{\|\mathbf{t} - \mathbf{x}\|_2 : \mathbf{x} \in \mathcal{L}\}$.

Hard Problem 2 — Bounded Distance Decoding (BDD)

Let n and m be positive integers and $\eta > 0$ be a parameter. The Bounded Distance Decoding problem is the following problem:

> **Given** a basis $(\mathbf{b}_0, \cdots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L} \subset \mathbb{R}^m$, and an element $\mathbf{t} \in \mathbb{R}^m$ that verifies $dist(\mathbf{t}, \mathcal{L}) \leq \eta \cdot \lambda_1(\mathcal{L})$
>
> **Find** a vector $\mathbf{x} \in \mathcal{L}$ closest to $\mathbf{t}$.

For high-dimensional lattices, the aforementioned uSVP and BDD problems are assumed hard even with the use of quantum computers. Contrary to the factorization and discrete logarithm problems, these lattice problems are supported by strong computational hardness properties. For example, on randomized instances, in certain ranges of parameters, the uSVP and BDD problems are NP-hard [Ajt98; Ajt96; Aro+93]. Besides, the security analysis and complexity estimation of both problems, in classical and quantum computation models, have been an active line of research during the past 20 years. In this thesis, we briefly present two algorithmic tools for the security analysis that will be used in the next chapters.

Babai rounding. Given a lattice $\mathcal{L}$ and a point $\mathbf{t}$, Babai's rounding algorithm is used to find a point $\mathbf{x} \in \mathcal{L}$ "reasonably close" to $\mathbf{t}$ [Bab85; Len81; LLL82]. It is presented in Algorithm 1. The $\lfloor \cdot \rceil$ represents a rounding to the nearest integer in $\mathbb{Z}$ applied coefficient-wise. Babai's algorithm is applied using the Moore-Penrose pseudoinverse $\mathbf{B}^{\sim} := \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ that verifies $\mathbf{B}\mathbf{B}^{\sim} = I_n$.

Algorithm 1 — Babai RoundOff algorithm

Data: a basis $(\mathbf{b}_0, \cdots, \mathbf{b}_{n-1}) \subset \mathbb{R}^m$ written as a matrix in $\mathbf{B} \in \mathbb{R}^{n \times m}$, $\mathbf{t} \in \mathbb{R}^m$
Result: $x \in \mathcal{L}$ close to $\mathbf{t}$

```
x := ⌊tB~⌉B
return x
```

Intuitively, the purpose of this rounding is to solve Hard Problem 2 by outputting a vector $\mathbf{x} \in \mathcal{L}$ close to $\mathbf{t}$.

Proposition 1. *Let $\mathbf{t} \in \mathbb{R}^m$, Algorithm 1 returns a vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{x}\|_2 \leq \frac{\sqrt{n}}{2} \cdot \|\mathbf{B}\|_{\text{sub}}$ where $\|\mathbf{B}\|_{\text{sub}} := \sup_{\mathbf{x} \in \mathbb{R}^m \setminus \{0\}} \frac{\|\mathbf{x}\mathbf{B}\|_2}{\|\mathbf{x}\|_2}$.*

Proof: The vector $\mathbf{x}$ is in the lattice as it is a linear combination of the lines of $\mathbf{B}$. Let $\mathbf{t}_0 := \mathbf{t}\mathbf{B}^{\sim}$, thus $\mathbf{t} = \mathbf{t}_0\mathbf{B}$ by definition of the Moore-Penrose pseudoinverse. Therefore,

$$\|\mathbf{t} - \mathbf{x}\|_2 = \|\mathbf{t}_0\mathbf{B} - \lfloor \mathbf{t}_0 \rceil \mathbf{B}\|_2 \leq \| \underbrace{\mathbf{t}_0 - \lfloor \mathbf{t}_0 \rceil}_{\in [-\frac{1}{2}, \frac{1}{2}]^n} \|_2 \cdot \|\mathbf{B}\|_{\text{sub}} .$$

□

As shown in Proposition 1, it is only efficient if the input basis $\mathbf{B}$ is given in a specific form for which $\|\mathbf{B}\|_{\text{sub}}$ is small, which is not the case for arbitrary bases. If, by chance, the vectors of the basis have a small norm and are almost orthogonal, $\|\mathbf{B}\|_{\text{sub}}$ will be small and Algorithm 1 can give accurate results.

The lattice reduction algorithms. As crucial part of lattice analysis, lattice reduction algorithms aim at computing a basis made of relatively short and almost-orthogonal vectors from an arbitrary input basis. A polynomial time algorithm was introduced by Lenstra, Lenstra,

and Lovasz (LLL) [LLL82] in 1982. It can output non-zero lattice vectors whose norms are greater than that of the shortest non-zero lattice vector by at most a factor $2^{O(n)}$. Even if this approximation seems bad for breaking any hard problem, LLL algorithm is still widely used in cryptanalysis.

A more recent lattice reduction algorithm called Block-Korkine-Zolotarev (BKZ) [Kan83; FP85; SE94] consists in applying LLL as a subroutine in projected sub-lattices with a fixed dimension β, called *blocksize*. BKZ offers trade-offs between the quality of the output and the complexity: higher blocksizes lead to better reduction quality (i.e. shorter vectors), at the expense of increasing the runtime. More exactly, the runtime increases exponentially in β. As discussed in [GN08; CN11; Alk+16b; Alb+17], predicting the quality of BKZ's output on a concrete lattice instance $\mathcal{L}$ is far from immediate and still under investigation. The current mostly used estimation makes the experimentally verified Geometric Series Assumption (GSA). We refer to [CN11] for the description and validity of this assumption. The estimation considers that, after applying BKZ with blocksize β (denoted BKZ$-\beta$ for short) the lattice basis is expected to contain a vector $\mathbf{b}$ that verifies

$$\|\mathbf{b}\|_2 = \delta_\beta^n \cdot \mathrm{Vol}(\mathcal{L})^{1/n}.$$

The factor δ_β is called *the root-Hermite factor* and for $\beta > 50$, it is predicted to follow

$$\delta_\beta \sim \left((\pi\beta)^{\frac{1}{\beta}} \cdot \frac{\beta}{2\pi e}\right)^{1/(2\beta-2)}. \tag{1.2}$$

Still assuming GSA, BKZ-β is able to solve uSVP (Hard Problem 1) instances whenever $\rho \geq \tau \cdot \delta_\beta^n$ where $\tau < 1$ is a constant depending on the lattice family. The BDD instances can also be transformed into uSVP ones and be solved with BKZ. We detail such a transformation in more details in Section 5.3.3. BKZ is currently the most utilized lattice reduction method [AKS01; HPS11b; HPS11a; MW15].

1.2.3 Other hard lattice-based problems

A second class of lattice problems arose from cryptographic constructions : Learning with Errors [Reg05] and Short Integer Solution [Ajt96]. They are somehow *ad-hoc* and thus cannot directly be seen as geometrically as the ones presented in Section 1.2.2.

Hard Problem 3 — Search learning with error (LWE)

Let n, m and q be positive integers, and let χ be a distribution over $\mathbb{Z}$.

Given the pair $\left(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{b} \coloneqq \mathbf{zA} + \mathbf{e} \in \mathbb{Z}_q^m\right)$ where:

1. $\mathbf{A}$ is sampled uniformly at random in $\mathbb{Z}_q^{n\times m}$,
2. $\mathbf{z}$ is sampled uniformly at random in $\mathbb{Z}_q^n$,
3. $\mathbf{e} \leftarrow \chi^m$.

Find $\mathbf{z}$ (or equivalently $\mathbf{e}$).

The hardness of Hard Problem 3 highly depends on the distribution χ. In fact, most of the hardness results consider that χ is a centered discrete Gaussian distribution with a small standard deviation (we refer to Section 2.2.2 for precisions on this distribution).

To relate LWE with uSVP, the authors of [Bra+13] prove that an LWE instance is as hard as standard geometric lattice problems.

Hard Problem 4 — Short Integer Solution (SIS)

Let n, m and q be positive integers and $\gamma > 0$ be a parameter.

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ sampled uniformly at random,

Find a non-zero vector $\mathbf{z} \in \mathbb{Z}_q^n$ such that

1. $\mathbf{zA} = 0$
2. $\|\mathbf{z}\|_2 \leq \gamma$

Similarly, the hardness of Hard Problem 4 highly depends on how small γ is and the underlying uSVP is simply the following: the solution $\mathbf{z}$ is a short vector in the lattice $\{\mathbf{x} | \mathbf{xA} = 0 \mod q\}$. The authors of [MR07] proved that Hard Problem 4 can be as hard as Section 1.2.2's problems.

1.2.4 Ring variants

For designing more efficient systems, ring variants of the previous problems were introduced. Although they were initially introduced for arbitrary polynomial rings, we instantiate a ring which will be used in the majority of the thesis. For n, q integer parameters such that $X^n + 1$ is irreducible in $\mathbb{Z}_q$, we define

$$R_q \coloneqq \frac{\mathbb{Z}_q[X]}{X^n + 1}. \tag{1.3}$$

Note that the parameter n does not appear in the notation as it is always a fixed parameter of the scheme. Throughout, we abuse notation and identify elements in $\mathbb{Z}_q$ with their representatives in $[-q/2, q/2)$, and elements in R_q with their representatives of degree $< n$. This allows us to define the ℓ_2-norm $\|\mathbf{x}\|_2$ of a polynomial $\mathbf{x} \in R_q$, so that $\|\mathbf{x}\|_2 \coloneqq \sqrt{\sum_i \mathbf{x}_i^2}$ where $\mathbf{x}_i \in [-q/2, q/2)$ are the coefficients of $\mathbf{x}$, and extend this to vectors of polynomials $\mathbf{y} \in R_q^l$ as $\|\mathbf{y}\|_2 \coloneqq \sqrt{\sum_i \|\mathbf{y}_i\|_2^2}$. Identically, we define and extend the ℓ_∞-norm.

The Module-LWE (or Mod-LWE) problem [LS12b] is a generalization of Hard Problem 3. It also includes the Ring-LWE problem introduced in [Ste+09; LPR10] as a particular case.

Hard Problem 5 — Search Module LWE (Mod-LWE)

Let n, m, q be parameters that define R_q as in Eq. (1.3), k be a positive integer and χ be a probability distribution over R_q

Given $(\mathbf{a}_i, b_i) \coloneqq (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{z} \rangle + e_i) \in R_q^k \times R_q$ for $i \in [0, m-1]$ where:

1. the $\mathbf{a}_i \in R_q^k$ are sampled uniformly at random for $i \in [0, m-1]$,
2. $\mathbf{z} \in R_q^k$ is sampled uniformly at random,
3. $\mathbf{e}_i \leftarrow \chi$ for $i \in [0, m-1]$.

Find $\mathbf{z}$ (or equivalently $\mathbf{e}$).

Ring-LWE is a particular case where $k = 1$.

Similarly, we present the module/ring variant of Hard Problem 4.

Hard Problem 6 — Module Short Integer Solution (Mod-SIS)

Let n, q be parameters that define R_q as in Eq. (1.3), k be a positive integer and $\gamma > 0$ be a parameter.

Given $\mathbf{a} \in R_q^k$ sampled uniformly at random,

Find a non-zero $\mathbf{z} \in R_q^k$ such that

1. $\langle \mathbf{z}, \mathbf{a} \rangle = 0$
2. for all coefficients $\mathbf{z}[i]$ of $\mathbf{z}$, $\forall i \in [0, k-1], \|\mathbf{z}[i]\|_\infty \leq \gamma$

Ring-SIS is a particular case where $k = 1$.

Under certain conditions, these ring variants can be as hard as the problems defined in Section 1.2.3 [LM06; PR06; Ste+09; LPR10; LPR13]. However, in practice their security is conjectured and it is somehow questioned because of the high structure of the underlying lattices. Some of the designs choose to avoid this ring structure [Bos+16].

Let us present another problem that was introduced before LWE and SIS [HPS98]. It is only defined with a cyclotomic ring and has been assumed hard since its introduction.

Hard Problem 7 — NTRU

Let n, q be two prime numbers that define R_q as in Eq. (1.3), and χ be a probability distribution over R_q.

Given $h = g/f \bmod q$ where f and g are sampled from χ.

Find g (or equivalently f).

1.2.5 Lattice constructions

In lattice-based cryptography, one advantage lies in the fact that the cryptosystems themselves have a remarkably simple description. The underlying elementary operations are often simple linear additions. Most of the work is in establishing and proving their security.

Public key encryption. Hoffstein, Pipher and Silverman developed one of the first constructions in 1998 [HPS98], they propose an efficient public key encryption based on Hard Problem 7, still assumed hard. This construction has inspired recent candidates for post-quantum standardization. Later, Regev suggested constructions using Hard Problem 3 in [Reg03; Reg06] and the first design has been introduced in [LP11]. The latter has also inspired very efficient recent candidates. We present in Table 1.1 lattice-based public key encryption schemes which may be standardized for post-quantum use.

Signatures. The first lattice-based signature scheme has been introduced in [GGH97] and the NTRU company followed by proposing NTRUSign [Hof+03; HPS01] based on Hard Problem 7. Both constructions were later broken because of a partial key leak in the distribution of the signatures [NR06]. Perturbation countermeasures were proposed by the authors of [Hof+03] but their security was subsequently broken in [DN12b]. Following this line of research, [GPV08] introduced a framework impervious to the latter statistical attacks. The distribution of its outputted signatures does not depend on the private key. This framework is at the origin of a family of lattice-based signatures called "hash-and-sign". It has led to a recent efficient candidate called Falcon [Pre+19] based on Hard Problem 7.

In parallel, another family of constructions emerged with Lyubashevsky's "Fiat–Shamir with aborts" paradigm [Lyu09] using Hards Problems 3 and 4. The underlying framework is called Fiat–Shamir *with aborts* because, unlike RSA and discrete logarithm-based constructions, lattice-based constructions involve sampling from sets that do not admit a nice algebraic structure. A naive sampling algorithm would leak partial key information, the same way as it did in schemes like GGH and NTRUSign; this is avoided by forcing the distribution of the output signature to be independent of the private key using rejection sampling. Many instantiations of the framework have been proposed [Lyu12; GLP12; Duc+13; PDG14; BG14] some of them very efficient. These works lead to many efficient post-quantum candidates.

Implementations of lattice-based signature schemes from both families are presented in Table 1.2.

Table 1.1: *Lattice-based public key encryption or key encapsulation mechanisms*

Name	Reference	Assumption	Mentioned in
NewHope	[Alk+16b; Alk+16a; Pöp+19]	Ring-LWE (Hard Problem 5)	Sections 4.2 and 5.6.2 and Remark 17
Frodo	[Bos+16; Nae+19]	LWE (Hard Problem 3)	Section 5.6.1
Crystals-Kyber	[Sch+19]	Mod-LWE (Hard Problem 5)	Section 4.3 and Remark 17
Saber	[DAn+19b]	Mod-LWR, variant of Mod-LWE	Section 4.3 and Remark 17
LAC	[Lu+19]	Ring-LWE (Hard Problem 5)	Remark 17, Perspective 5, and Section 5.6.3
NTRUprime	[Ber+17b; Ber+19]	NTRU (Hard Problem 7)	Remark 29
NTRU	[Zha+19]	NTRU (Hard Problem 7)	Section 5.6.3
Round5	[Gar+19; Baa+19]	GLWR, variant of Mod-LWE	Section 5.6.3

Table 1.2: *Lattice-based signature schemes*

Name	Reference	Family	Assumption	Mentioned in
GLP	[GLP12]	Fiat–Shamir with aborts	DCK, variant of Ring-LWE	Section 3.3
BLISS	[Duc+13]	Fiat–Shamir with aborts	Ring-SIS (Hard Problem 6)	Sections 2.4 and 3.4
Crystals-Dilithium	[Duc+18; Lyu+19]	Fiat–Shamir with aborts	Mod-SIS (Hard Problem 6)	Section 3.3.6
qTesla	[Alk+15; Alk+17; Bin+19]	Fiat–Shamir with aborts	Ring-LWE (Hard Problem 5)	Section 3.5
Falcon	[Pre+19]	Hash-and-Sign	NTRU (Hard Problem 7)	Section 2.5

1.3 Goal and contribution of the thesis

To address the shortcomings of existing cryptosystems and help build quantum-safe algorithms, the main goal of this thesis is the analysis of the real-world security of lattice-based post-quantum asymmetric schemes. Real-world in this context means that, in addition to the algorithms being selected as to offer a high conjectured black-box security in both classical and quantum computation models, the implementations should be efficient and secure for use in software, embedded systems, and in hardware.

In other words, in addition to black-box cryptanalysis, we intend to investigate the security of lattice-based cryptosystems in *extended security models*. It implies studying {classical, quantum, misuse, timing, physical}-attacks on lattice-based schemes and possible combinations. With this wide program, we focus, in particular, on the possibility of *leveraging and addressing the new features introduced by the lattice structures*. These new features will be encountered and detailed later in the thesis. They include for instance: rejection sampling for signatures (detailed in Section 2.2.5), use of transcendental functions (detailed in Section 2.3), use of Gaussian distributions (presented in Section 2.2.4), non-perfect correctness for public key encryption (presented in Section 4.1.1), private keys structured in large vectors of small elements and reliance on lattice reduction algorithms. With this thesis, we aim at:

- on the one hand, constructing provable algorithmic countermeasures when it is possible;
- on the other hand, estimating precisely the applicability and complexity of attacks in order to provide practical security assessments of the corresponding implementations.

Part I Constructions. Concerning physical and timing attacks, the cryptographic research community made numerous contributions on the possibility of applying such attacks to lattice-based schemes [ADP18a; Bru+16; GP18; Rav+19; Bos+18a; CMP18]. To continue the effort towards efficient and secure implementations, we put on our designer's hat and investigate the algorithmic protections against timing and side-channel attacks along with proofs of impermeability against these attacks. This part focuses on *signature* schemes because they are more subject to attacks and their protection seem challenging due to hard-to-protect features like rejection sampling. We first focus on introducing algorithmic tools to tackle these features. Next, we apply them to several schemes. In Table 1.3, we outline the main literature gaps that joint research contributions presented in this thesis aim at filling in terms of protection of known lattice-based signatures schemes.

Table 1.3: *Constructive contributions of this thesis*

Signature scheme	Provable timing protection Chapter 2	Provable high-order masking Chapter 3
GLP	[GLP12]	this thesis Section 3.3
BLISS	this thesis Section 2.4	this thesis Section 3.4
Falcon	this thesis Section 2.5	see Perspective 4
Crystals-Dilithium	[Lyu+19]	[Mig+19]
qTesla	[Bin+19]	this thesis Section 3.5

Part II Cryptanalysis. Concerning less studied attacks like quantum or misuse attacks, the lattice-based schemes are neither trivially vulnerable nor provably secure. This thesis takes a pass on improving the complexity assessment in various attack scenarios.

- Chapter 4. A potential source of weakness lies in the lack of perfect correctness of lattice-based schemes. Our work participates in showing that it may enable decryption failures attacks. In a first step, we introduce a misuse attack inspired from [Flu16]. Later, based

on the work of [DAn+19a], we introduce an improved generic quantum attack targeting public key encryptions. The security is then estimated as a function of the maximum number of decryption queries and the (quantum or classical) work necessary.

- Chapter 5. We introduce an algorithmic tool to assess the security of the mathematical problem underlying a lattice-based scheme when the scheme is the target of *a combination of attacks* (classical and side-channel for example). This tool automatically builds lattice reduction attacks and estimates their cost when hints of various forms are given. In other words, this tool allows to assess the new security of the scheme *after* attacks have been performed and a partial knowledge of the secret is available. The usefulness of the presented framework is demonstrated with the improvement of several known attacks.

Parts I and II concern different cryptosystems and notions. They can be read independently.

1.4 Publications

1.4.1 Constructive results

- *Masking the GLP Lattice-Based Signature Scheme at Any Order* [Bar+18]. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi and Mehdi Tibouchi.

  This paper, whose results are presented in Chapter 3, proposes the first provable algorithmic countermeasure against physical attacks for a lattice-based signature. This countermeasure has been previously applied to the decryption procedure of some lattice-based encryption schemes, but the much more difficult case of signatures (which are highly non-linear and typically involve randomness) were not considered until this paper. We focus on the GLP scheme of Güneysu, Lyubashevsky and Pöppelmann [GLP12]. We show how to provably mask it in the Ishai–Sahai–Wagner model [ISW03] at any order in a relatively efficient manner, using extensions of the techniques of Coron et al. for converting between arithmetic and Boolean masking [Cor17]. Our proof relies on a mild generalization of probing security that supports the notion of public outputs. We also provide a proof-of-concept implementation to assess the efficiency of the proposed countermeasure. This paper has been published in the proceedings of the conference EUROCRYPT in 2018.

- *An Efficient and Provable Masked Implementation of qTESLA.* [GR19] François Gérard and Mélissa Rossi.

  The results of this paper are also presented in Chapter 3. In this paper, we study the lattice-based signature scheme qTESLA in the context of the masking countermeasure. Continuing the line of research opened by the previous paper [Bar+18] with the masking of the GLP signature scheme, we extend and modify it to mask qTESLA. Based on the work of Migliore et al. [Mig+19], we slightly modify the parameters to improve the masked performance while keeping the same security. The masking can be done at any order and specialized gadgets are used to get maximal efficiency at order 1. We implemented our countermeasure in the original code of the submission and performed tests at different orders to assess the feasibility of our technique. This paper has been published in the proceedings of the conference CARDIS in 2019.

- *GALACTICS: Gaussian Sampling for Lattice-Based Constant-Time Implementation of Cryptographic Signatures, Revisited* [Bar+19a]. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi and Mehdi Tibouchi.

  The results of this paper are presented in Chapters 2 and 3. We propose a constant-time implementation of the BLISS lattice-based signature scheme. The outstanding performance of the BLISS signature scheme stems in large part from its reliance on discrete Gaussian distributions, which allows for better parameters and security reductions. However, that advantage has also proved to be its Achilles' heel, as discrete Gaussians pose serious challenges in terms of secure implementations. Implementations of BLISS so far have included secret-dependent branches and memory accesses, both as part of the discrete Gaussian sampling and of the essential rejection sampling step in signature generation. These defects have led to multiple devastating timing attacks, and were a key reason why BLISS was not submitted to the NIST postquantum standardization effort. In fact, almost all of the actual candidates chose to stay away from Gaussians despite their efficiency advantage, due to the serious concerns surrounding implementation security. Moreover, simple countermeasures will often not cut it: a first contribution – not presented in this thesis as it was handled by Thomas Espitau and Mehdi Tibouchi– shows that a reasonable-looking countermeasure suggested in previous work to protect the BLISS rejection sampling can again be defeated using novel timing attacks, in which the timing information is fed to a phase retrieval machine learning algorithm in order to achieve a full key recovery. Fortunately, as a second contribution, we present careful implementation techniques that allow us to describe an implementation of BLISS with complete timing attack protection, achieving the same level of efficiency as the original unprotected code, without resorting on floating point arithmetic or platform-specific optimizations like AVX intrinsics. These techniques, including a new approach to the polynomial approximation of transcendental functions, can be applied to the masking of the BLISS signature scheme, and will hopefully make more efficient and secure implementations of lattice-based cryptography possible going forward. This paper has been published in the proceedings of the conference ACM-CCS in 2019.

- *Isochronous Gaussian Sampling: From Inception to Implementation* [How+20] with James Howe, Thomas Prest, Thomas Ricosset and Mélissa Rossi.

  The results of this paper are also presented in Chapter 2. Gaussian sampling over the integers is a crucial tool in lattice-based cryptography, but has proven over the recent years to be surprisingly challenging to perform in a generic, efficient and provable secure manner. In this paper, we present a modular framework for generating discrete Gaussians with arbitrary center and standard deviation. Our framework is extremely simple, and it is precisely this simplicity that allowed to make it easy to implement, provably secure, portable, efficient, and provably resistant against timing attacks. Our sampler is a good candidate for any trapdoor sampling and it is actually the one that has been recently implemented in the Falcon signature scheme. Our second contribution – not presented in this thesis as it was mainly handled by another co-author James Howe – aims at systematizing the detection of implementation errors in Gaussian samplers. We provide a statistical testing suite for discrete Gaussians called SAGA (Statistically Acceptable GAussian). In a nutshell, our two contributions take a step towards trustable and robust Gaussian sampling real-world implementations. This paper has been published in the proceedings of the conference PQ-CRYPTO in 2020.

1.4.2 Cryptanalysis results

- *Assessment of the Key-Reuse Resilience of NewHope* [Bau+19]. Aurélie Bauer, Henri Gilbert, Guénaël Renault and Mélissa Rossi.

  The results of this paper are presented in Chapter 4. NewHope is a suite of two efficient Ring-LWE based key encapsulation mechanisms (KEMs) that has been proposed to the NIST call for proposals for post-quantum standardization. In this paper, we study the security of NewHope when an active adversary accesses a key establishment and is given access to an oracle, called key mismatch oracle, which indicates whether her guess of the shared key value derived by the party targeted by the attack is correct or not. This attack model turns out to be relevant in key reuse situations since an attacker may then be able to access such an oracle repeatedly with the same key either directly or using faults or side channels, depending on the considered instance of NewHope. Following this model we show that, by using NewHope recommended parameters, several thousands of queries are sufficient to recover the full private key with high probability. This result has been experimentally confirmed using Magma CAS implementation. While the presented key mismatch oracle attacks do not break any of the designers' security claims for the NewHope KEMs, they provide better insight into the resilience of these KEMs against key reuse. In the case of the CPA-KEM instance of NewHope, they confirm that key reuse (e.g. key caching at server side) should be strictly avoided, even for an extremely short duration. In the case of the CCA-KEM instance of NewHope, they allow to point out critical steps inside the CCA transform that should be carefully protected against faults or side channels in case of potential key reuse. This paper has been published in the proceedings of the conference CT-RSA in 2019.

- *(One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes* [DRV20]. Jan-Pieter D'Anvers, Mélissa Rossi and Fernando Virdia.

  The results of this paper are also presented in Chapter 4. Lattice-based encryption schemes are often subject to the possibility of decryption failures, in which valid encryptions are decrypted incorrectly. Such failures, in large number, leak information about the private key, enabling an attack strategy alternative to pure lattice reduction. Extending the "failure boosting" technique of D'Anvers et al. in PKC 2019, we propose an approach that we call "directional failure boosting" that uses previously found "failing ciphertexts" to accelerate the search for new ones. We analyse in detail the case where the lattice is defined over polynomial ring modules quotiented by $\langle X^n + 1 \rangle$ and give a proof of concept on a simple Mod-LWE-based scheme parametrized à la Kyber768/Saber. We show that, using our technique, for a given private key (single-target setting), the cost of searching for additional failing ciphertexts after one or more have already been found, can be sped up dramatically. We thus demonstrate that, in this single-target model, these schemes should be designed so that it is hard to even obtain one decryption failure. Besides, in a wider security model where there are many target private keys (multi-target setting), our attack greatly improves over the state of the art. This paper has been published in the proceedings of the conference EUROCRYPT in 2020.

- *LWE with Side Information: Attacks and Concrete Security Estimation* [Dac+20]. Dana Dachman-Soled, Léo Ducas, Huijing Gong and Mélissa Rossi.

  The results of this paper are presented in Chapter 5. We propose a framework for cryptanalysis of lattice-based schemes, when side information — in the form of "hints" — about the secret and/or error is available. Our framework generalizes the so-called primal lattice

reduction attack, and allows the progressive integration of hints before running a final lattice reduction step. Our techniques for integrating hints include sparsifying the lattice, projecting onto and intersecting with hyperplanes, and/or altering the distribution of the secret vector. Our main contribution is to propose a toolbox and a methodology to integrate such hints into lattice reduction attacks and to predict the performance of those lattice attacks with side information. While initially designed for side-channel information, our framework can also be used in other cases: exploiting decryption failures, or simply exploiting constraints imposed by certain schemes (LAC, Round5, NTRU), that were previously not known to (slightly) help lattice attacks. We implement a Sage 9.0 toolkit to actually mount such attacks with hints when computationally feasible, and to predict their performances on larger instances. We provide several end-to-end application examples, such as an improvement of a single trace attack on Frodo by Bos et al (SAC 2018). Contrary to ad-hoc practical attacks exploiting side-channel leakage, our work is a generic way to estimate security loss even given very little side-channel information. This paper has been accepted to the conference CRYPTO in 2020.

1.4.3 Other cryptanalysis publications

As the subject of this thesis originally included code-based and multivariate cryptography, we also studied cryptanalytic tools for these structures. In particular, we used Gröbner basis techniques for a student project with Goldreich pseudorandom generator.

- *A side-channel assisted cryptanalytic attack against QcBits* [Ros+17]. Mélissa Rossi, Mike Hamburg, Michael Hutter and Mark E. Marson.

  QcBits is a code-based public key algorithm based on a problem thought to be resistant to quantum computer attacks. It is a constant time implementation for a quasi-cyclic moderate density parity check (QC-MDPC) Niederreiter encryption scheme, and has excellent performance and small key sizes. In this paper, we present a side-channel key-recovery attack against QcBits. We first used differential power analysis (DPA) against the syndrome computation of the decoding algorithm to recover partial information about one half of the private key. We then used the recovered information to set up a system of noisy binary linear equations. Solving this system of equations gave us the entire key. Finally, we propose a simple but effective countermeasure against the power analysis used during the syndrome calculation. This paper has been published in the proceedings of the conference CHES in 2017.

- *On the Concrete Security of Goldreich's Pseudorandom Generator* [Cou+18]. Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi and Yann Rotella.

  Local pseudorandom generators allow to expand a short random string into a long pseudorandom string, in such a way that each output bit depends on a constant number d of input bits. Due to its extreme efficiency features, this intriguing primitive enjoys a wide variety of potential applications in cryptography and complexity. In the polynomial regime, where the seed is of size n and the output of size n^s for $s > 1$, the only known solution, commonly known as Goldreich's PRG, proceeds by applying a simple d-ary predicate to public random size-d subsets of the bits of the seed. While the security of Goldreich's PRG has been thoroughly investigated (with a variety of results deriving provable security guarantees against classes of attacks in some parameter regimes and necessary criteria to be satisfied by the underlying predicate), little is known about its concrete security and efficiency. Motivated by its numerous theoretical applications and the hope of getting

practical instantiations for some of them, we initiate a study of the concrete security of Goldreich's PRG, and evaluate its resistance to cryptanalytic attacks. Along the way, we develop a new guess-and-determine-style attack, and identify new criteria which refine existing criteria and more accurately capture the security guarantees of candidate local PRGs. This paper has been published in the proceedings of the conference ASIACRYPT in 2018.

Part I

Constructive results

Chapter

2 Protecting lattice-based signatures against timing attacks

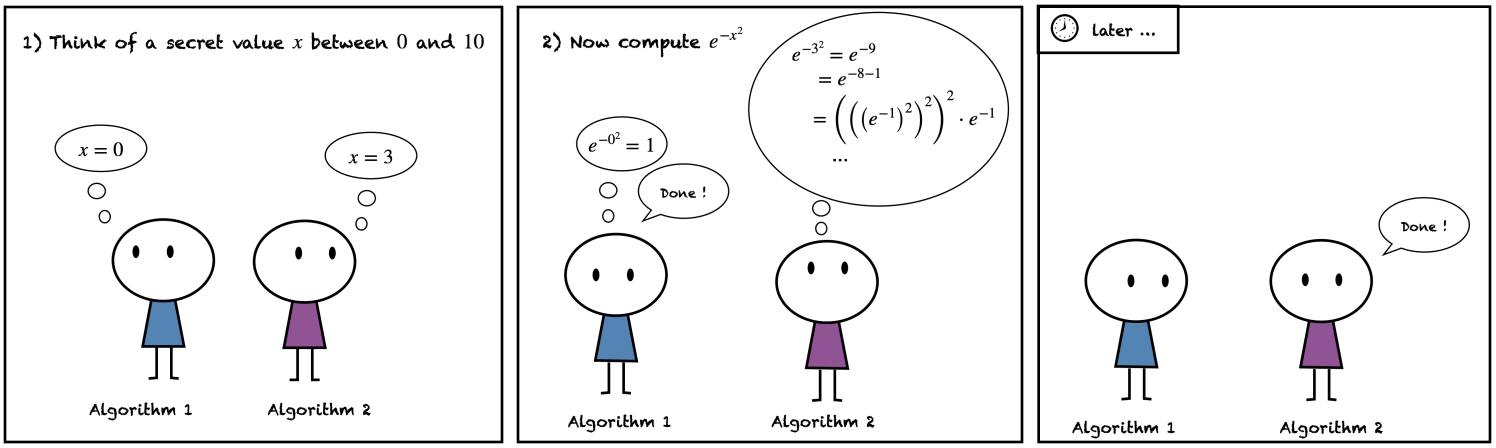


Figure 2.1: *Timing attacks consist in analyzing the time that an algorithm takes to make its computations. In this example, the timing corresponds to the number of multiplications while computing* $\exp(-x^2)$. *When* $x = 0$ *(for the blue algorithm), there is no multiplication. When* $x = 3$ *(for the purple one), an exponentiation should be done and several multiplications are necessary. Thus, the timing is correlated to the secret value* x. *The purpose of this chapter is to modify the algorithms so that the timing does not depend on the secret value* x.

This chapter describes *how to protect lattice-based signature schemes against timing attacks*. It tackles the problem of Gaussian distributions. This work began with the latmasking working group (Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque and Mehdi Tibouchi) because we were not able to mask a lattice-based signature called BLISS [Duc+13]. In fact, there was a slight timing leakage in the Gaussian sampling, as presented in the last attacked and protected version [Esp+17]. Interestingly, we were able to kill both timing and side-channel breaches with one stone: the tools we made to handle Gaussians in order to mask BLISS can also be applied for provably protecting BLISS against timing attacks. It has been gratifying designing the polynomial approximation presented in Section 2.3 with Mehdi Tibouchi's precious help. The purpose was to use simple functional analysis techniques to show that Gaussian distributions are not that annoying, as they can be transformed into polynomials. Finally, Mehdi Tibouchi implemented the resulting protected scheme to show that the protections could be almost inexpensive. In addition to other contributions (like a timing attack highlighting the timing leakage performed by Mehdi Tibouchi and Thomas Espitau), this work was part of [Bar+19a] which was accepted to the ACM-CCS conference en 2019.

The techniques used in this previous work could be reused for Falcon's Gaussian sampler too. Thus, by joining an ongoing project with Thomas Prest – my Thales advisor at that time – and Thomas Ricosset, we were able to formalize Falcon Gaussian sampler and proved its security against timing attacks. We finally figured that this sampler could also be made generic to be reused in other settings. We sent our preliminary work to Thomas Pornin, who implemented our

Gaussian sampler into the official implementation of Falcon. The security/performance tradeoff eventually achieved is very satisfying, as it does not harm Falcon's ranking in the NIST standardization process. James Howe eventually joined us for writing a paper combining our sampler and his statistical test suite for Gaussians. The final work was published in 2020 [How+20].

Hopefully, thanks' to these simple techniques (polynomial approximations, Rényi divergence arguments, convolution, acceptance-rejection lemma), the Gaussian distributions can be somehow reconsidered for practical implementations as they also provide better-suited parameters.

Chapter content

2.1 Introduction and motivations

Gaussian distributions and implementation vulnerabilities. Despite their attractive theoretical properties, lattice-based constructions present novel challenges in terms of implementation security, particularly concerning their protection against side-channel attacks. Taking signatures as an example, possibly one of the most efficient constructions is the BLISS signature scheme of Ducas et al. [Duc+13], which features excellent performance and has seen real-world deployment via the VPN software suite strongSwan. Later implementations of BLISS show good hardware performance as well [PDG14]. However, existing implementations of BLISS suffer from

significant leakage through timing side-channels, which have led to several devastating attacks against the implementations [Bru+16; PBY17; Esp+17; Boo+18; TW19]. The main feature of BLISS exploited in these attacks is the use of discrete *Gaussian distributions*, either as part of the Gaussian sampling used to generate the random nonces in signatures or as part of the crucial rejection sampling step that forms the core of the Fiat–Shamir with aborts framework that supports BLISS security.

Generally speaking, Gaussian distributions are ubiquitous in theoretical lattice-based cryptography, thanks to their convenient behavior concerning proofs of security and parameter choices. However, their role in practical implementations is less clear, mainly because of the concerns surrounding implementation attacks. For this reason, some schemes limit or proscribe the use of Gaussians [Bin+19; Lyu+19]. For example, BLISS was not submitted to the NIST post-quantum standardization effort partly due to those concerns. Besides, the second round candidate Dilithium [Duc+18], which can be seen as a direct successor of BLISS, replaces Gaussian distributions by uniform ones, at the cost of larger parameters and a less efficient implementation, specifically citing implementation issues as their justification.

However, in some situations, Gaussians are unavoidable. The most prominent example is trapdoor sampling [GPV08; Pei10; MP12]: performing it efficiently with other distributions is an open question, except in limited cases [LW15] which entail a growth $O(\sqrt{n})$ to $O(n)$ of the output, resulting in dwindling security levels. Given the countless applications of trapdoor sampling (full-domain hash signatures [GPV08; Pre+19], identity-based encryption (or IBE) [GPV08; DLP14], hierarchical IBE [Cas+10; ABB10], etc.), it is important to come up with Gaussian samplers over the integers, which are not only efficient, but also provably secure, resistant to timing attacks, and in general easy to deploy.

Signature security property

All the recent signature schemes are claimed EUF-CMA (existential unforgeability under chosen message attack). Let us introduce the definition.

Adversary		**Challenger**
	$\xleftarrow{(\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})}$	
	$\xleftarrow{pk}$	$(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathtt{Q}_\mathrm{s}$ queries	$\xrightarrow{m^{(1)}}$	
		$\sigma^{(1)} \leftarrow \mathsf{Sign}(sk, m^{(1)})$
	$\xleftarrow{\sigma^{(1)}}$	
	$\vdots$	
	$\xrightarrow{m^{(\mathtt{Q}_\mathrm{s})}}$	
		$\sigma^{(\mathtt{Q}_\mathrm{s})} \leftarrow \mathsf{Sign}(sk, m^{(\mathtt{Q}_\mathrm{s})})$
	$\xleftarrow{\sigma^{(\mathtt{Q}_\mathrm{s})}}$	
forgery	$\xrightarrow{m^*, \sigma^*}$	$b \coloneqq \mathsf{Verify}(pk, m^*, \sigma^*) \wedge (m^* \notin \{m^{(1)}, \ldots, m^{(\mathtt{Q}_\mathrm{s})}\})$

Security Game 1: *EUF-CMA game.*

Security Model 1. *Let* $\mathtt{Q}_\mathrm{s}$ *be a fixed maximum amount of signature queries. A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is EUF-CMA-secure if any probabilistic polynomial time adversary has a negligible probability of winning in the EUF-CMA game presented in Security Game 1.*

This security definition does not capture any timing attacker. The goal of the following work is to design a stronger security property combining EUF-CMA and timing guarantees.

2.2 Definitions and tools

2.2.1 Isochrony

Let us now formalize the timing security guarantees that a signature implementation can claim. To do so, we introduce the notion of *isochronous* existential unforgeability under chosen message attack (I-EUF-CMA), which combines the standard EUF-CMA security property with the timing security.

> After discussing with Thomas Prest, Thomas Ricosset and James Howe, we concluded that the term "isochronous" should be preferred to "constant-time". Indeed, the timing security is ensured if the execution time *does not* depend on the private key, may it vary. An algorithm can run in variable time and still be impervious to timing attacks.

Security Model 2 (Isochrony)**.** *An implementation* (*KeyGen*, *Sign*, *Verify*) *of a signature scheme is isochronously EUF-CMA-secure, or I-EUF-CMA secure for short if any probabilistic polynomial time adversary has a negligible winning probability in the experiment from Security Game 2. In that security experiment,* *ExecObs* *takes as input an algorithm and its arguments, executes the program, and outputs the result of the computation together with the timing leakage* $\mathscr{L}^{(i)}_{\mathsf{Sign}}$*, consisting of a measurement of the execution time of the* i*-th execution of the algorithm* Sign*.*

Adversary		**Challenger**
	$\xleftarrow{(\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})}$	
	$\xleftarrow{pk}$	$(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathsf{Q_s}$ queries	$\xrightarrow{\mu^{(1)}}$	$\left(\sigma^{(1)}, \mathscr{L}^{(1)}_{\mathsf{Sign}}\right) \leftarrow \mathsf{ExecObs}(\mathsf{Sign}, \mu^{(1)}, sk)$
	$\xleftarrow{\sigma^{(1)}, \mathscr{L}^{(1)}_{\mathsf{Sign}}}$	
	$\vdots$	
	$\xrightarrow{\mu^{(\mathsf{Q_s})}}$	$\left(\sigma^{(\mathsf{Q_s})}, \mathscr{L}^{(\mathsf{Q_s})}_{\mathsf{Sign}}\right) \leftarrow \mathsf{ExecObs}(\mathsf{Sign}, \mu^{(\mathsf{Q_s})}, sk)$
	$\xleftarrow{\sigma^{(\mathsf{Q_s})}, \mathscr{L}^{(\mathsf{Q_s})}_{\mathsf{Sign}}}$	
forgery	$\xrightarrow{\mu^*, \sigma^*}$	$b \coloneqq \mathsf{Verify}(pk, \mu^*, \sigma^*) \wedge (\mu^* \notin \{\mu^{(1)}, \ldots, \mu^{(\mathsf{Q_s})}\})$

Security Game 2: *The I-EUF-CMA security game.* *ExecObs* *is a universal Turing machine that takes as input an algorithm and several possible arguments, and it returns the output of the algorithm on those arguments, together with the timing leakage* $\mathscr{L}$*.*

In the following sections, we use some assumptions to ensure isochrony provably. First, we will study the distribution of the execution time as a function of elementary operations. We thus need to consider that the latter cannot leak through timing.

Assumption 1. The elementary operations $\{+, -, \times, /\}$ over integer and floating-point numbers are assumed to be performed in an isochronous way: their execution time is independent of their inputs.

The previous assumption can be validated if the elementary operations are performed with specific certified isochronous instructions. With general CPU instructions, [Por20] showed that Assumption 1 is not always verified.

2.2.2 Gaussians

For $\sigma, \mu \in \mathbb{R}$ with $\sigma > 0$, we call Gaussian function of parameters σ, μ and denote by $\rho_{\sigma,\mu}$ the function defined over $\mathbb{R}$ as $\rho_{\sigma,\mu}(x) \coloneqq \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. Note that when $\mu = 0$ we omit it in index notation, i.e. $\rho_\sigma(x) \coloneqq \rho_{\sigma,0}(x)$. The parameter σ (resp. μ) is often called the standard deviation (resp. center) of the Gaussian. In addition, for any countable set $S \subsetneq \mathbb{R}$ we abusively denote by $\rho_{\sigma,\mu}(S)$ the sum $\sum_{z\in S} \rho_{\sigma,\mu}(z)$ when the latter converges.

In all this thesis, we denote by $D_{S,\sigma,\mu}$ and call discrete Gaussian distribution of parameters σ, μ the distribution over S defined by $D_{S,\sigma,\mu}(z) \coloneqq \rho_{\sigma,\mu}(z)/\rho_{\sigma,\mu}(S)$. Here too, when $\mu = 0$ we omit it in index notation, e.g. $D_{S,\sigma}(z) \coloneqq D_{S,\sigma,0}(z)$. We also use the notation $\mathcal{B}_p$ to denote the Bernoulli distribution of parameter p

We define the support of a probability distribution, denoted $Supp(\cdot)$, as the set of elements with nonzero probability.

2.2.3 Generalized Rényi divergence results

We present the definition of the Rényi divergence, which we will use massively in our security proofs. Lemma 2 and its proof are a formalization based on the work of [Pre17, Section 3.3]. The generalization to arbitrary security parameters in Lemma 4 is also a slight contribution of this thesis.

Definition 1 (Rényi Divergence)**.** *Let $\mathcal{P}$, $\mathcal{Q}$ be two distributions such that $Supp(\mathcal{P}) \subseteq Supp(\mathcal{Q})$. For $a \in (1, +\infty)$, we define the Rényi divergence of order a by*

$$R_a(\mathcal{P}, \mathcal{Q}) \coloneqq \left(\sum_{x \in Supp(\mathcal{P})} \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

In addition, we define the Rényi divergence of order $+\infty$ by

$$R_\infty(\mathcal{P}, \mathcal{Q}) \coloneqq \max_{x \in Supp(\mathcal{P})} \frac{\mathcal{P}(x)}{\mathcal{Q}(x)}.$$

The Rényi divergence *is not a distance*; for example, it is neither symmetric nor does it verify the triangle inequality, which makes it less convenient than the statistical distance. On the other hand, it does verify cryptographically useful properties, including a few listed below.

In this thesis, random variables, i.e. variables whose values depend on outcomes of a random phenomenon, are denoted in lowercase calligraphic letters e.g. $\mathfrak{a}, \mathfrak{b}, \mathfrak{x}$. Random vectors are denoted in uppercase calligraphic letters e.g. $\mathfrak{C}, \mathfrak{X}, \mathfrak{Z}$.

Lemma 1 ([Bai+15])**.** *For two distributions $\mathcal{P}, \mathcal{Q}$ and two finite families of distributions $(\mathcal{P}_i), (\mathcal{Q}_i)$, the Rényi divergence verifies these properties:*

- ***Data processing inequality.*** *For any function f, $R_a(f(\mathcal{P}), f(\mathcal{Q})) \leq R_a(\mathcal{P}, \mathcal{Q})$ where the distribution $f(\mathcal{P})$ (resp. $f(\mathcal{Q})$) is obtained by applying f to $x \leftarrow \mathcal{P}$ (resp. $x \leftarrow \mathcal{Q}$).*

- ***Probability preservation.*** *For any event $E \subseteq Supp(\mathcal{Q})$ and $a \in (1, +\infty)$,*

$$\mathcal{Q}(E) \geq \mathcal{P}(E)^{\frac{a}{a-1}} / R_a(\mathcal{P}, \mathcal{Q}), \tag{2.1}$$
$$\mathcal{Q}(E) \geq \mathcal{P}(E) / R_\infty(\mathcal{P}, \mathcal{Q}). \tag{2.2}$$

- ***Multiplicativity.*** *Assume that $\mathcal{P}$ and $\mathcal{Q}$ are the joint distribution of a pair of random variables $(\mathfrak{x}_1, \mathfrak{x}_2)$. Assume also that $\mathfrak{x}_1$* ***and*** *$\mathfrak{x}_2$* ***are independent****. We then denote $\mathcal{P}(\mathfrak{x}_1, \mathfrak{x}_2) \coloneqq \mathcal{P}_1(\mathfrak{x}_1) \cdot \mathcal{P}_2(\mathfrak{x}_2)$ and $\mathcal{Q}(\mathfrak{x}_1, \mathfrak{x}_2) \coloneqq \mathcal{Q}_1(\mathfrak{x}_1) \cdot \mathcal{Q}_2(\mathfrak{x}_2)$. Then, $R_a(\mathcal{P}, \mathcal{Q}) = R_a(\mathcal{P}_1, \mathcal{Q}_1) \cdot R_a(\mathcal{P}_2, \mathcal{Q}_2)$.*

Application for security arguments. A first demonstrated in [Bai+15] and later in [Pre17, Section 3.3], the Rényi divergence has proved to be convenient for security arguments in lattice-based cryptography. We introduce two general lemmas (Lemmas 2 and 4) that will be used for security arguments in this chapter. In some cases, the queries to the distributions are not independent[1], thus, one cannot apply the multiplicativity directly in our proofs. To handle these dependencies, we introduce the following proposition.

Proposition 2. *Let $\mathcal{P}$ and $\mathcal{Q}$ denote two distributions of a N-uple of random variables $(\mathfrak{x}_i)_{i<N}$. For $0 \leq i < N$, assume $\mathcal{P}_i$ (resp $\mathcal{Q}_i$) is the marginal distribution of X_i, and let $\mathcal{P}_{i|<i}(\cdot|x_{<i})$ denote the conditional distribution of $\mathfrak{x}_i$ given that $(\mathfrak{x}_0, \ldots, \mathfrak{x}_{i-1}) = x_{<i}$. Let $a > 1$. Suppose that for all $0 \leq i < N$, there exists $r_{a,i} \geq 1$ such that for all i-uple $x_{<i}$ in the support of $\mathcal{Q}$ restricted to its first i variables,*

$$R_a(\mathcal{Q}_{i|x_{<i}}, \mathcal{P}_{i|x_{<i}}) \leq r_{a,i}. \tag{2.3}$$

Then:

$$R_a(\mathcal{Q}, \mathcal{P}) \leq \prod_{i<N} r_{a,i}. \tag{2.4}$$

Proof: We prove the result in the case $N = 2$, the general case then follows by induction. We have:

$$R_a(\mathcal{Q}, \mathcal{P})^{a-1} = \sum_{x_1, x_2} \frac{(\mathcal{Q}_1(x_1) \cdot \mathcal{Q}_{2|1}(x_2|x_1))^a}{(\mathcal{P}_1(x_1) \cdot \mathcal{P}_{2|1}(x_2|x_1))^{a-1}} \tag{2.5}$$

$$= \sum_{x_1} \frac{\mathcal{Q}_1(x_1)^a}{\mathcal{P}_1(x_1)^{a-1}} \cdot \left(\underbrace{R_a\left(\mathcal{Q}_{2|1}(\cdot|x_1), \mathcal{P}_{2|1}(\cdot|x_1)\right)}_{\leq r_{a,2}} \right)^{a-1} \tag{2.6}$$

$$\leq (r_{a,1} \cdot r_{a,2})^{a-1} \tag{2.7}$$

This concludes the proof. □

Lemma 2. *Consider a cryptosystem denoted $C_\mathcal{P}$ in which a $\mathtt{Q}$-uple $(x_0, \ldots x_{\mathtt{Q}-1})$ is drawn according to a distribution $\mathcal{P}$. Assume that $C_\mathcal{P}$ is λ-bit secure against a search problem. Let us consider a copy of the latter cryptosystem, denoted $C_\mathcal{Q}$, where the only difference is that $(x_0, \ldots x_{\mathtt{Q}-1})$ is drawn according to a distribution $\mathcal{Q}$ such that $Supp(\mathcal{Q}) \subseteq Supp(\mathcal{P})$.*

We denote $(\mathfrak{x}_0, \ldots \mathfrak{x}_{\mathtt{Q}-1})$ the random variables corresponding to $(x_0, \ldots x_{\mathtt{Q}-1})$ that follow $\mathcal{P}$ or $\mathcal{Q}$ depending on the case. For $0 \leq i < \mathtt{Q}$, we denote by $\mathcal{P}_{i|<i}(\cdot|x_{<i})$ (resp. $\mathcal{Q}_{i|<i}(\cdot|x_{<i})$) the conditional distribution of $\mathfrak{x}_i$ given the knowledge of $(\mathfrak{x}_1, \ldots, \mathfrak{x}_{i-1}) = x_{<i}$ under $\mathcal{P}$ (resp. $\mathcal{Q}$). If, for all $0 \leq i < \mathtt{Q}$, and for all i-uple $x_{<i}$ in the support of $\mathcal{Q}$ restricted to its first i variables,

$$R_{2\lambda-1}(\mathcal{Q}_{i|x_{<i}}, \mathcal{P}_{i|x_{<i}}) \leq 1 + \frac{1}{4\mathtt{Q}}, \tag{2.8}$$

then $C_\mathcal{Q}$ is $(\lambda - 1)$-bit secure against the search problem.

[1] We thank Damien Stehlé for pointing that out.

Note that when the drawing of x_i is independent from the preceding drawings, Eq. (2.8) is equivalent to assume for all $0 \leq i < \mathtt{Q}$,

$$R_{2\lambda-1}(\mathcal{Q}_i, \mathcal{P}_i) \leq 1 + \frac{1}{4\mathtt{Q}}. \tag{2.9}$$

When they are not independent, we informally assume that the Rényi divergence is small no matter the preceding knowledge.

Proof: Let us assume that $C_{\mathcal{Q}}$ is *not* $(\lambda-1)$-bit secure against the search problem. Then, we assume that an attacker that breaks the scheme $C_{\mathcal{Q}}$ by solving the search problem with a probability, denoted $\varepsilon_{\mathcal{Q}}$, that is higher than $2^{-(\lambda-1)}$ exists. Thus,

$$\varepsilon_{\mathcal{Q}} > 2^{-(\lambda-1)}.$$

To succeed, the attacker draws a $\mathtt{Q}$-uple $(x_0, \dots x_{\mathtt{Q}-1})$ (in $Supp(\mathcal{Q})$) according to the distribution $\mathcal{Q}$. Let us denote $\varepsilon_{\mathcal{P}}$ the probability that the same attacker breaks the scheme $C_{\mathcal{P}}$ with querying the distribution $\mathcal{P}$ instead of $\mathcal{Q}$. By probability preservation, for all $a \in (1, +\infty)$,

$$\varepsilon_{\mathcal{P}} \geq \varepsilon_{\mathcal{Q}}^{a/(a-1)} / R_a(\mathcal{Q}, \mathcal{P}) \tag{2.10}$$

Let us set $a := 2\lambda - 1$ in Eq. (2.10). For the numerator, we have

$$\varepsilon_{\mathcal{Q}}^{a/(a-1)} > \left(2^{-(\lambda-1)}\right)^{\frac{2\lambda-1}{2(\lambda-1)}} = \sqrt{2} \cdot 2^{-\lambda}. \tag{2.11}$$

Thus, $\varepsilon_{\mathcal{P}}$ satisfies

$$\varepsilon_{\mathcal{P}} > \frac{\sqrt{2} \cdot 2^{-\lambda}}{R_{2\lambda-1}(\mathcal{Q}, \mathcal{P})} \tag{2.12}$$

We apply Proposition 2 using the hypothesis in Eq. (2.8),

$$R_{2\lambda-1}(\mathcal{Q}, \mathcal{P}) \leq \left(1 + \frac{1}{4\mathtt{Q}}\right)^{\mathtt{Q}} \leq e^{1/4} \leq \sqrt{2}. \tag{2.13}$$

Thus,

$$\varepsilon_{\mathcal{P}} > 2^{-\lambda}. \tag{2.14}$$

We can conclude that our attacker is also able to break $C_{\mathcal{P}}$ by solving the search problem with a success probability strictly higher than $2^{-\lambda}$.

Thus, assuming that $C_{\mathcal{Q}}$ is *not* $(\lambda-1)$-bit secure against the search problem leads to proving that $C_{\mathcal{P}}$ is *not* λ-bit secure against the search problem either. The lemma is, therefore, proved by contraposition.

□

Relative error lemma The following lemma, also introduced by Prest in [Pre17], shows that a bound of δ on the relative error between two distributions implies a bound $O(a\delta^2)$ on the log of the Rényi divergence (as opposed to a bound $O(\delta)$ on the statistical distance).

Lemma 3 (Lemma 3 of [Pre17])**.** *Let $\mathcal{P}, \mathcal{Q}$ be two distributions such that $Supp(\mathcal{P}) = Supp(\mathcal{Q})$. Suppose that the relative error between $\mathcal{P}$ and $\mathcal{Q}$ is bounded:*

$$\exists \delta > 0 \text{ such that } \forall x \in Supp(\mathcal{Q}) \quad \left|\frac{\mathcal{Q}(x)}{\mathcal{P}(x)} - 1\right| \leq \delta.$$

Then, for all $a \in (1, +\infty)$:

$$R_a(\mathcal{Q}, \mathcal{P}) \leq \left(1 + \frac{a(a-1)\delta^2}{2(1-\delta)^{a+1}}\right)^{\frac{1}{a-1}} \underset{\delta \to 0}{\sim} 1 + \frac{a\delta^2}{2}$$

Let us now introduce a new security theorem that reformulates Lemma 2 with a different "close enough" condition that is expressed in terms of relative error using Lemma 3.

Lemma 4. *Consider a cryptosystem denoted $C_{\mathcal{P}}$ in which a $\mathtt{Q}$-uple $(x_0, \dots x_{\mathtt{Q}-1})$ is drawn according to a distribution $\mathcal{P}$. Assume that $C_{\mathcal{P}}$ is λ-bit secure against a search problem. Let us consider a copy of the latter cryptosystem, denoted $C_{\mathcal{Q}}$, where the only difference is that $(x_0, \dots x_{\mathtt{Q}-1})$ is drawn according to a distribution $\mathcal{Q}$ such that $Supp(\mathcal{Q}) = Supp(\mathcal{P})$.*

We denote $(\mathfrak{x}_0, \dots \mathfrak{x}_{\mathtt{Q}-1})$ the random variables corresponding to $(x_0, \dots x_{\mathtt{Q}-1})$ that follow $\mathcal{P}$ or $\mathcal{Q}$ depending on the case. For $0 \leq i < \mathtt{Q}$, we denote by $\mathcal{P}_{i|<i}(\cdot|x_{<i})$ (resp. $\mathcal{Q}_{i|<i}(\cdot|x_{<i})$) the conditional distribution of $\mathfrak{x}_i$ given the knowledge of $(\mathfrak{x}_1, \dots, \mathfrak{x}_{i-1}) = x_{<i}$ under $\mathcal{P}$ (resp. $\mathcal{Q}$). If, for all $0 \leq i < \mathtt{Q}$ and for all i-uple $x_{<i}$ in the support of $\mathcal{Q}$ restricted to its first i variables, $\mathcal{P}_{i|x_{<i}}$ and $\mathcal{Q}_{i|x_{<i}}$ satisfy

$$\forall x \in Supp(\mathcal{P}) \quad \left| \frac{\mathcal{P}_{i|x_{<i}}(x) - \mathcal{Q}_{i|x_{<i}}(x)}{\mathcal{P}_{i|x_{<i}}(x)} \right| \leq \frac{1}{\sqrt{2\,(2\lambda - 1) \cdot \mathtt{Q}}}, \tag{2.15}$$

then $C_{\mathcal{Q}}$ is $(\lambda - 1)$-bit secure against the search problem.

Proof: Let us fix $0 \leq i < \mathtt{Q}$ and a i-uple $x_{<i}$ in the support of $\mathcal{Q}$ restricted to its first i variables. Eq. (2.15) can be rewritten as follows.

$$1 - \delta \leq \frac{\mathcal{Q}_{i|x_{<i}}}{\mathcal{P}_{i|x_{<i}}} \leq 1 + \delta \text{ with } \delta := \frac{1}{\sqrt{2\,(2\lambda - 1) \cdot \mathtt{Q}}}.$$

Since $Supp(\mathcal{Q}_{i|x_{<i}}) = Supp(\mathcal{P}_{i|x_{<i}})$ (because $Supp(\mathcal{P}) = Supp(\mathcal{Q})$), we apply $\mathtt{Q}$ times Lemma 3 with $a = 2\lambda - 1$ and get

$$\begin{aligned} R_{2\lambda-1}(\mathcal{Q}_{i|x_{<i}}, \mathcal{P}_{i|x_{<i}}) &\leq 1 + \frac{(2\lambda-1)\cdot\frac{1}{2(2\cdot\lambda-1)\mathtt{Q}}}{2} \\ &= 1 + \tfrac{1}{4\mathtt{Q}}. \end{aligned}$$

The application of Lemma 2 allows to conclude. □

2.2.4 Gaussian sampling

The purpose of Gaussian sampling is to perform the instruction $x \leftarrow D_{\mathbb{Z},\sigma,\mu}$ where $D_{\mathbb{Z},\sigma,\mu}$ is a discrete Gaussian with parameters σ and μ as defined in Section 2.2.2. The latter are either parameters of the cryptosystem or intermediate variables.

Several works devoted to sampling according to discrete Gaussian distributions [MW17; Kar+18; DN12a] or related distributions, such as rounded Gaussians [HLS18] exist. Different methods exist, according to the standard deviation size, as well as whether it is constant or variable. Encryption schemes typically require small standard deviations, while signatures use larger ones, which are fixed for Fiat–Shamir schemes and vary for hash-and-sign constructions. It is customary to use a small standard deviation "base sampler" and build upon it to achieve the desired large standard deviation: this is the approach presented in [MW17]. These works can also be distinguished according to whether they require floating-point arithmetic. In particular, the rounded Gaussians of [HLS18] offer numerous attractive properties, but they have some statistical limitations in terms of distinguishing advantage, and they rely on floating-point implementations. We present here the isochronous Gaussian Sampling building blocks that we will use in this thesis.

Optimization with positive samples

Before showing the sampling techniques, one optimization could be to sample from $|D_{\mathbb{Z},\sigma}|$ where the latter denotes the distribution of the absolute value of the samples from $D_{\mathbb{Z},\sigma}$. Therefore, to sample from $D_{\mathbb{Z},\sigma}$, one must use a random sign flip after drawing from $|D_{\mathbb{Z},\sigma}|$ to recover the entire distribution $D_{\mathbb{Z},\sigma}$.

Remark 1. *One can remark that* $|D_{\mathbb{Z},\sigma}| \neq D_{\mathbb{Z}^+,\sigma}$. *Indeed* $D_{\mathbb{Z}^+,\sigma}(z) = \rho_\sigma(z)/\rho_\sigma(\mathbb{Z}^+)$ *while*

$$|D_{\mathbb{Z},\sigma}|(z) \propto \begin{cases} 2 \cdot \rho_\sigma(z) & \text{if } z \neq 0 \\ \rho_\sigma(z) & \text{if } z = 0 \end{cases}$$

Isochronous Cumulative Distribution Table (CDT)

Let us assume that we want to draw a sample from a distribution D. In particular; one might take $D := |D_{\mathbb{Z},\sigma}|$ for a direct Gaussian sampling with a sign generation just after as will be presented later in Chapter 3 with Gadget 11. Alternatively, one might take $D = D_{\mathbb{Z}^+,\sigma}$ for using different techniques as will be presented in Algorithms 12 and 13.

In any case, to draw from D, the general inverse transform sampling technique originally introduced in [Dev86, Section 2.2] can be used. It needs a Cumulative Distribution Table (CDT for short): precompute a table of the cumulative distribution function of D. The table should cover the interval at the points where the distribution has a non-negligible probability $\gtrsim 2^{-128}$. Then, to produce a sample; one can generate a random value in $[0, 1]$ with 128 bits of precision, and return the index of the first entry in the table strictly larger than that value. In variable time, this can be done relatively efficiently with a binary search, but this leaks the resulting sample through memory access patterns. As a result, an isochronous implementation has essentially no choice but to read the entire table each time and carry out each and every comparison. This process is summed up in Algorithm 2. The parameter w is the number of elements of the CDT and θ is the precision of its coefficients. The CDT table contains w elements t_j for $j \in [0, w-1]$ such that $t_j := \sum_{0 \leq i \leq j} D(i)$ with θ bits of absolute precision.

Algorithm 2 — Cumulative Distribution Table Sampling (SampleCDT)

Params: w, θ, $(t_j)_{j \in [0,w-1]}$
Result: A sample z following D

$z := 0$
pick $u \in [0, 1)$ uniformly with θ bits of absolute precision
for $0 \leq j < w$ **do**
  $b := (t_j \leq u)$ `/* b = 1 if (t_j ≤ u), b = 0 otherwise */`
  $z := z + b$
end
return z

Although a basic CDT implementation would store the cumulative probabilities with 128 bits of precision, it is, in fact, possible to only store lower precision approximations, as discussed in [PDG14; Wal19] (see also [Pre17] for an alternate approach using "conditional distribution functions"). Nevertheless, even with these optimizations, a direct CDT leads to highly inefficient implementations because the tables are often very heavy. Besides, this sampling does not handle variable center and standard deviation.

Generic Convolution techniques

An efficient approach, originally introduced by Pöppelmann et al. [PDG14] and later improved and generalized by Micciancio and Walter [MW17], assumes that we can generate a base Gaussian distribution $D_{\mathbb{Z}^+,\sigma_0}$ with not too small standard deviation σ_0 using an isochronous CDT for example. And, this allows combining samples issued with the base distribution to achieve larger standard deviations. These convolution techniques are highly generic and can be applied for any variable center and standard deviation. Inevitably, this convolution technique is no always optimal for practical schemes with particular specificities. For example, for BLISS, the fixed large standard deviation can be leveraged (see Section 2.4.5). Furthermore, for our sampler for Falcon in Section 2.5, we leverage the fact that the variation magnitude of the center and standard deviation is very small. Both of the specific techniques use rejection sampling, explained below.

2.2.5 Rejection sampling

Let us present the acceptance-rejection method [CRW04; MM10; Lyu12; Duc+13]. This technique, that allows to convert a set of samples from random instances of a family g_v of distributions into a subset of samples from a target distribution f, is based on the following lemma.

Lemma 5 (Acceptance-Rejection Lemma as stated in [Duc+13])**.** *Let V be an arbitrary set, $h : V \to \mathbb{R}$ and $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. If $g_v : \mathbb{Z}^m \to \mathbb{R}$ is a family of distributions indexed by $v \in V$ with the following property*

$$\exists M \in \mathbb{R} \text{ such that } \forall v \in V \quad \forall \mathbf{z} \in \mathbb{Z}^m \quad M g_v(\mathbf{z}) \geq f(\mathbf{z}).$$

The distribution of the output of the following procedure

1. $v \leftarrow h$
2. $\mathbf{z} \leftarrow g_v$
3. *output* $(\mathbf{z}, v)$ *with probability* $\frac{f(\mathbf{z})}{M g_v(\mathbf{z})}$

is identical to the output of

1. $v \leftarrow h$
2. $\mathbf{z} \leftarrow f$
3. *output* $(\mathbf{z}, v)$ *with probability* $1/M$.

A first use of Lemma 5 is the possibility to eliminate secret-dependent output distributions with a secret-independent number of iterations as sketched in Section 2.2.6. Another key idea of the rejection sampling is the possibility to build efficient Gaussian sampling by

1. constructing a distribution that looks somewhat like the desired Gaussian but is not statistically close;
2. using rejection sampling to correct the discrepancy.

This is particularly useful when step 1 can be performed efficiently. This idea is applied to both our Gaussian sampler for BLISS signature scheme in Section 2.4 and our isochronous sampler for Falcon in Section 2.5.

2.2.6 An example of application

Let us consider the Fiat-Shamir with aborts signature scheme introduced in [Lyu12]. This scheme is the ancestor of BLISS (described in Section 3.4), Dilithium (see Section 3.3.6) and qTesla (see Section 3.5). This scheme is based on SIS (see Hard Problem 4) and we present the key generation and signature in Algorithms 3 and 4. We refer to [Lyu12] for the parameter

Algorithm 3 — Lyubashevsky key generation

Result: Private key $sk = \mathbf{S}$, Public key $pk = (\mathbf{A}, \mathbf{T})$

$\mathbf{S} \xleftarrow{\$} \{-d, \dots, d\}^{m \times k}$

$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$

$\mathbf{T} \coloneqq \mathbf{AS}$

Algorithm 4 — Lyubashevsky signature

Data: m, pk, sk

Result: Signature sig

$\mathbf{y} \leftarrow D_{\mathbb{Z},\sigma}^m$

$\mathbf{u} \coloneqq \mathbf{Ay}$

$\mathbf{c} \coloneqq \mathsf{hash}(\mathbf{u}, m)$

$\mathbf{z} \coloneqq \mathbf{y} + \mathbf{sc}$

return sig $\coloneqq (\mathbf{z}, \mathbf{c})$ with probability $\frac{\rho_\sigma(\|\mathbf{z}\|_2)}{M\rho_\sigma(\|\mathbf{z}-\mathbf{Sc}\|_2)}$

Algorithm 5 — Hybrid signature algorithm

Data: m, pk, sk

Result: Signature sig

$\mathbf{c} \xleftarrow{\$} \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$

$\mathbf{z} \leftarrow D_{\mathbb{Z},\sigma}^m$

return sig $\coloneqq (\mathbf{z}, \mathbf{c})$ with probability $\frac{1}{M}$ and program $\mathsf{hash}(\mathbf{Az} - \mathbf{Tc}, m) = \mathbf{c}$

choice and verification algorithm. The hash function is chosen to satisfy $\mathsf{hash} : \{0,1\}^* \to \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^k, \|\mathbf{v}\|_1 \leq \kappa\}$.

Thanks to the rejection sampling lemma (we refer to [Lyu12, Theorem 5.1] for the detailed proof), the signature algorithm is indistinguishable from the hybrid presented in Algorithm 5.

Remark 2. *The number of iterations is variable and follows a geometric distribution of parameter M. Note that even though the number of iterations varies, it is **independent from the secret**. This stays true even if an attacker is able to select a subset of signatures by filtering*[2] *the output $\mathbf{c}$. Indeed, assume that an attacker has access to the expectation of rejection conditionned with $\mathbf{c}$, i.e. $\mathbb{E}_{\mathbf{y}}[(\mathbf{z}, \mathbf{c})$ rejected $\mid \mathbf{c}]$. The following computation shows that this quantity is independent from $\mathbf{S}$ and $\mathbf{c}$. Let $\mathbf{S}$ and $\mathbf{c}$ be fixed quantities,*

$$p_{sc} \coloneqq \mathbb{E}_{\mathbf{y}}[(\mathbf{z}, \mathbf{c}) \text{ rejected} \mid \mathbf{c}] = 1 - 1/M \cdot \mathbb{E}_{\mathbf{y}}\left[\frac{\rho_\sigma(\|\mathbf{z}\|_2)}{\rho_\sigma(\|\mathbf{z} - \mathbf{Sc}\|_2)}\right] \tag{2.16}$$

$$= 1 - 1/M \cdot \mathbb{E}_{\mathbf{y}}\left[\frac{\rho_\sigma(\|\mathbf{y} + \mathbf{sc}\|_2)}{\rho_\sigma(\|\mathbf{y}\|_2)}\right] \tag{2.17}$$

$$= 1 - \frac{1}{M\rho_\sigma(\mathbb{Z}^n)} \sum_{\mathbf{y}} \frac{\rho_\sigma(\|\mathbf{y}\|_2)\rho_\sigma(\|\mathbf{y} + \mathbf{sc}\|_2)}{\rho_\sigma(\|\mathbf{y}\|_2)} \tag{2.18}$$

$$= 1 - \frac{1}{M\rho_\sigma(\mathbb{Z}^n)} \sum_{\mathbf{y}} \rho_\sigma(\|\mathbf{y} + \mathbf{sc}\|_2) \tag{2.19}$$

$$= 1 - \frac{1}{M\rho_\sigma(\mathbb{Z}^n)} \sum_{\mathbf{z}} \rho_\sigma(\|\mathbf{z}\|_2) \tag{2.20}$$

$$= 1 - 1/M. \tag{2.21}$$

One may argue that, by fixing $\mathbf{c}$, we also filter elements $\mathbf{y}$ though the hashing function and thus the sum will not be on all $\mathbf{y}$. However, in the random oracle model, the choices of $\mathbf{c}$ and $\mathbf{y}$ can be seen as independents as long as $(\mathbf{u}, m)$ are not repeated which happens with negligible probability.

[2] We thank Damien Stehlé for suggesting this approach.

2.3 GALACTICS: A polynomial approximation tool

This section, along with Section 2.4, are part of a work [Bar+19a] done with Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque and Mehdi Tibouchi. The purpose of this section is to use Lemma 4 for replacing transcendental functions with finite input set (the density function of an ideal distribution $\mathcal{P}$) by polynomials with the same support.
The well-studied techniques for approximating transcendental functions with polynomials have become very relevant in implementations of Gaussian sampling algorithms. They are in particular useful as the $\exp(\cdot)$, and $\cosh(\cdot)$ functions are necessary for Gaussian and Bimodal Gaussian distributions. One first naive technique could be to use a Taylor development or the transcendental function. However, one can find lower degrees and get a higher accuracy. For example, in [Pre17] Prest suggests a Gaussian sampler that uses Padé approximants to evaluate the exponential function of the rejection probability. More recently, Zhao et al. have proposed a polynomial approximation using the Sollya software package [SMC10]. Their resulting implementation runs without floating-point division [ZSS18] since that operation is known to leak through timing. Zhao et al. use floating-point multiplications instead to compute the exponential, but this instruction does not always have isochronous execution guarantees either. Here, we approximate the transcendental functions over an interval using integer polynomials to avoid floating operations. Moreover, we aim to approximate using polynomials with small coefficients so that we can use small-sized integers and obtain a straightforward implementation of Horner's polynomial evaluation algorithm.

Definition 2 (Polynomial Approximation)**.** *Let $K \in \mathbb{R}^+$, for f a nonzero function on an interval I, we denote by P_f^I a polynomial in $\mathbb{R}[x]$ that satisfies*

$$\forall x \in I, \quad \left| \frac{P_f^I(x) - f(x)}{f(x)} \right| < 2^{-K} \tag{2.22}$$

Such a polynomial is referred to as an approximation that coincides with f up to K bits of relative precision on I.

Lemma 6. *Let $f : I \to \mathbb{R}$ be a function defined on an interval $I \subset \mathbb{R}$ such that $\forall x \in I\ 0 \leq f(x) \leq 1$. Consider a cryptosystem denoted $\mathcal{C}$ that requires drawing a $\mathtt{Q}$-uple $(a_0, \dots a_{\mathtt{Q}-1}) \in \{0,1\}^{\mathtt{Q}}$ such that $a_i \leftarrow \mathcal{B}_{f(x_i)}$ where the x_i are drawn from a distribution on I. Assume that $\mathcal{C}$ is λ-bit secure against a search problem. Let*

$$K := \log_2 \left(\sqrt{2 \cdot (2 \cdot \lambda - 1) \cdot \mathtt{Q}} \right).$$

Let us consider a copy of the latter cryptosystem, denoted C_{approx}, where the only difference is that each $a_i \leftarrow \mathcal{B}_{f(x_i)}$ is replaced by $a_i \leftarrow \mathcal{B}_{P_f^I(x_i)}$ for the same x_i where P_f^I is such that

$$\forall x \in I, \left| \frac{P_f^I(x) - f(x)}{f(x)} \right| < 2^{-K} \tag{2.23}$$

and

$$\forall x \in I \left| \frac{P_f^I(x) - f(x)}{1 - f(x)} \right| \leq 2^{-K}. \tag{2.24}$$

Then, $\mathcal{C}_{\text{approx}}$ is $(\lambda - 1)$-bit secure against the search problem.

Proof: For all $x \in I$,

$$\left| \frac{\mathcal{B}_{f(x)}(1) - \mathcal{B}_{P_f^I(x)}(1)}{\mathcal{B}_{f(x)}(1)} \right| = \left| \frac{P_f^I(x) - f(x)}{f(x)} \right| \text{ and } \left| \frac{\mathcal{B}_{f(x)}(0) - \mathcal{B}_{P_f^I(x)}(0)}{\mathcal{B}_{f(x)}(0)} \right| = \left| \frac{P_f^I(x) - f(x)}{1 - f(x)} \right|.$$

$$\text{Thus, } \forall b \in \{0,1\} \quad \left| \frac{\mathcal{B}_{f(x)}(b) - \mathcal{B}_{P_f^I(x)}(b)}{\mathcal{B}_{f(x)}(b)} \right| \leq 2^{-K} = \frac{1}{\sqrt{2\,(2\lambda-1)\cdot \mathtt{Q}}}.$$

Using Lemma 4, one can safely replace the distribution $\mathcal{B}_{f(x)}$ by the distribution $\mathcal{B}_{P_f^I(x)}$ and the resulting cryptosystem will be $(\lambda-1)$-bit secure against the search problem. □

Remark 3. *The condition of Eq. (2.24) has been added in a recent modification of this thesis. Thus, in our applications, Eq. (2.24) will be checked experimentally once a polynomial is found. The next paragraphs focus on a polynomial approximation verifying Eq. (2.23).*

In this section, we aim at exhibiting a polynomial P_f^I that validates Definition 2 with implementation friendly features, i.e. short-sized coefficients and a short degree. More precisely, we must minimize two parameters in order to achieve Eq. (2.22) as tightly as possible, namely

- η, the number of bits of its coefficients,
- γ, its degree.

The procedure is as follows:

1. In a first attempt, we exhibit a candidate polynomial for f whose coefficients are in the field $\mathbb{R}$. This step gives us the minimum degree γ that is needed.
2. In a second attempt, the coefficients of the candidate polynomial are rounded to fulfill the requirement on the number of bits η.

2.3.1 Polynomial approximation in $\mathbb{R}[x]$

We start by looking for a polynomial $P_{\mathbb{R}}$ in $\mathbb{R}[x]$ that approximates f on a defined interval I such that $\forall x \in I, f(x) \leq 1/2$. In this section, we re-define the infinite norm as $\|f\|_\infty \coloneqq \sup_I |f(x)|$. Such a bound is convenient as Eq. (2.22) is equivalent to

$$\left\| \frac{P_f^I}{f} - 1 \right\|_\infty < 2^{-K}. \tag{2.25}$$

However, the main drawback of the infinite norm is that it is not Euclidian. One possible efficient method to polynomially approximate with the infinite norm is introduced in [BC18]. In the following, we present a different method which trades efficiency with accuracy: the approximation consists of the interpolation of a continuous interval instead of a discrete set of samples. This procedure is more adapted to our setting since we want an approximation for all x in an interval I.

To approximate on I, we can over-approximate the infinite norm by a Euclidian norm in order to use orthogonal projections. We then propose the use of Sobolev H^2 inner product. This Euclidean metric was introduced in [Sob63] and allowed to upper bound the infinite norm. Let us describe this inner product.

Definition 3 (Sobolev H^2 inner product)**.** *For u and v two differentiable functions defined on an interval I, Sobolev H^2 inner product is defined by*

$$\langle u, v \rangle_S \coloneqq \frac{1}{|I|} \int_I uv + |I| \int_I u'v'.$$

The corresponding norm $|\cdot|_S$ is

$$|u|_S^2 \coloneqq \frac{1}{|I|} \int_I u^2 + |I| \int_I u'^2.$$

And the interesting property of this norm is the following.

Lemma 7 ([Kel20]). *The Sobolev norm $|\cdot|_S$ satisfies*

$$\|u\|_\infty \leq \sqrt{2} \cdot |u|_S .$$

Proof: Let $x_0 \in I$ be such that $|u(x)| \geq |u(x_0)|$ for all $x \in I$. We then write

$$u(x) = u(x_0) + \int_{x_0}^{x} u'$$

Hence,

$$|u(x)| \leq |u(x_0)| + \int_{x_0}^{x} |u'| \leq \frac{1}{|I|} \int_I |u| + \int_I |u'|$$

Using Cauchy-Schwarz, we have $\left(\int_I |u|\right)^2 \leq |I| \int_I |u|^2$ and $\left(\int_I |u'|\right)^2 \leq |I| \int_I |u'|^2$. Then,

$$|u(x)| \leq \frac{1}{|I|} \sqrt{|I| \int_I |u|^2} + \sqrt{|I| \int_I |u'|^2}$$

Using the equality $x + y \leq \sqrt{2}\sqrt{x^2 + y^2}$ for $x, y \geq 0$, we have

$$|u(x)| \leq \sqrt{2} \sqrt{\frac{1}{|I|^2} \cdot |I| \int_I |u|^2 + |I| \int_I |u'|^2}$$

Thus,

$$||u||_\infty \leq \sqrt{2} \cdot |u|_{\mathrm{S}}$$

which concludes the proof. □

Based on this norm, we compute a polynomial, denoted $P_{\mathbb{R}}$, as an orthogonal projection minimizing $|P_{\mathbb{R}}/f - 1|_{\mathrm{S}}$ and get

$$\left\| \frac{P_{\mathbb{R}}}{f} - 1 \right\|_\infty \leq \sqrt{2} \cdot \underbrace{\left| \frac{P_{\mathbb{R}}}{f} - 1 \right|_{\mathrm{S}}}_{\text{small}} .$$

Let $\mathbb{R}_{<d}[X]$ be the space of polynomials of degree $< d$. To find an element of the form $\frac{P_{\mathbb{R}}}{f}$ that minimizes $\left|\frac{P_{\mathbb{R}}}{f} - 1\right|_{\mathrm{S}}$, we build the following vector space:

$$E_d := \left\{ X \mapsto \frac{P(X)}{f(X)} : P \in \mathbb{R}_{<d}[X] \right\}$$

Assuming that f is a transcendental function, the function $X \mapsto 1$ does not belong in E_d for any d. However, for each d, there exists an element of the form $\frac{P_{\mathbb{R}}}{f} \in E_d$ that minimizes $\left|\frac{P_{\mathbb{R}}}{f} - 1\right|_{\mathrm{S}}$. The latter element corresponds to the orthogonal projection of 1 on E_d with respect to Sobolev H^2 inner product.

For each d, we thus derive an orthonormal basis of E_d by Gram-Schmidt orthonormalizing the canonical basis $\left(1/f, X/f, \cdots, X^{d-1}/f\right)$ with respect to $|\cdot|_{\mathrm{S}}$; getting a basis denoted $(g_0, \cdots, g_{d-1})$. Let π_d denote the projection of 1 on E_d,

$$\pi_d = \textstyle\sum_{i=0}^{d-1} \langle 1, g_i \rangle_S \, \frac{g_i}{|g_i|_{\mathrm{S}}}. \tag{2.26}$$

With Lemma 7, $\|\Pi_d - 1\|_\infty \leq \sqrt{2} \cdot |\Pi_d - 1|_S$. With the application of the log function, a slightly underestimated quality of the approximation can be obtained as

$$\kappa(d) := -\log_2 \left(\sqrt{2} \cdot \left|\Pi_d - 1\right|_S\right). \tag{2.27}$$

Therefore, to achieve a precision K, as shown in Algorithm 6, it is sufficient to select the degree γ as being the minimum degree such that

$$\kappa(\gamma) > K, \text{ i.e. } \|\Pi_\gamma - 1\|_\infty < 2^{-K}, \text{ and set } P_{\mathbb{R}} = \Pi_\gamma \cdot f.$$

Algorithm 6 — Polynomial Approximation with large coefficients (floatapprox)

Params: The function f, the interval I
Data: The target relative precision K
Result: A polynomial $P_{\mathbb{R}}$ verifying Eq. (2.25) along with its degree

$d := 1$
repeat
  $\Pi_d :=$ Projection of $x \mapsto 1$ on E_d with respect to $|.|_S$
  $\kappa(d) := -\log_2 \left(\sqrt{2} \cdot \left|\Pi_d - 1\right|_S\right)$
  $d := d + 1$
until $\kappa \geq K$
return $P_{\mathbb{R}} := \Pi_d \cdot f$ and $\gamma := d$

2.3.2 Integer polynomial approximation

Let us assume that we have found a polynomial $P_{\mathbb{R}} \in \mathbb{R}[X]$

$$P_{\mathbb{R}}(X) = a_0 + a_1 \cdot X + \cdots + a_\gamma X^\gamma \tag{2.28}$$

with $(a_i)_{i\in[0,\gamma]} \in \mathbb{R}$ verifying Eq. (2.25). We now want to derive P_f^I from $P_{\mathbb{R}}$ with coefficients of the following form.

$$P_f^I(X) = b_0 \cdot 2^{-m_0} + b_1 \cdot 2^{-m_1} \cdot X + \cdots + b_\gamma \cdot 2^{-m_\gamma} \cdot X^\gamma, \tag{2.29}$$

where $(m_i)_{i\in[0,\gamma]}$ are integers and $(b_i)_{i\in[0,\gamma]}$ are η-bit signed integers (i.e. $\in [-2^{\eta-1}, 2^{\eta-1}[$).

In order to obtain these coefficients, we need to operate a rounding of $P_{\mathbb{R}}$ and minimize the precision loss. An intuitive idea could be to multiply each coefficient a_i by a large power of two and round it to the nearest integer. Actually, one can lose less precision by using classical lattice reduction techniques.

In a nutshell, the idea is to round $P_{\mathbb{R}}$ with its closest element in a Euclidean lattice that represents the elements in $\mathbb{Z}_{2^\eta}[X]/f(X)$. In this objective, for each coefficient a_i, we first fix m_i to the unique integer such that

$$a_i \cdot 2^{m_i} \in [-2^{\eta-1}, 2^{\eta-1}[. \tag{2.30}$$

Let us create an Euclidean lattice with the following basis

$$B := \left(2^{-\eta-m_i} \cdot \frac{X^i}{f(X)}\right)_{i\in[0,\gamma]}. \tag{2.31}$$

Our notion of closeness still refers to the Sobolev norm, which is an unusual norm for Euclidean lattices. The lattice reduction must be adapted to use the Sobolev norm (using Gram matrix corresponding to Sobolev's inner product). And, this lattice can be LLL-reduced[3] with respect to the Sobolev H^2 norm. A Babai rounding of the polynomial $P_{\mathbb{R}}/f$ with respect to the same Sobolev H^2 norm gives candidates coefficients for P_f^I. The quality of the rounding can be evaluated as

$$\kappa^{\text{round}}(\gamma,\eta) := -\log_2\left(\sqrt{2}\cdot\left|\frac{P_f^I}{f}-\frac{P_{\mathbb{R}}}{f}\right|_{\mathrm{S}}\right). \tag{2.32}$$

The rounding always implies a loss of accuracy in the approximation. One can compute the relative precision of P_f^I by using Eqs. (2.27) and (2.32)

$$K' := 2^{-\kappa(\gamma)} + 2^{-\kappa^{\text{round}}(\gamma,\eta)}.$$

If $K' > K$, then P_f^I verifies Eq. (2.22), i.e. P_f^I is a correct approximation that coincides with f up to K bits of relative precision on I. Indeed, in that case:

$$\begin{aligned}
\forall x \in I,\quad \frac{|P_f^I(x)-f(x)|}{|f(x)|} &\le \left\|\frac{P_f^I-f}{f}\right\|_\infty = \left\|\frac{P_f^I}{f}-1\right\|_\infty \\
&\le \left\|\frac{P_f^I}{f}-\frac{P_{\mathbb{R}}}{f}\right\|_\infty + \left\|\frac{P_{\mathbb{R}}}{f}-1\right\|_\infty \\
&\le \sqrt{2}\cdot\left|\frac{P_f^I}{f}-\frac{P_{\mathbb{R}}}{f}\right|_{\mathrm{S}} + \sqrt{2}\cdot\left|\frac{P_{\mathbb{R}}}{f}-1\right|_{\mathrm{S}} \\
&= 2^{-\kappa(\gamma)} + 2^{-\kappa^{\text{round}}(\gamma,\eta)} \\
&\le 2^{-K}.
\end{aligned}$$

Algorithm 7 — Sketch of the full polynomial approximation

Params: The function f, the interval I
Data: The target relative precision K, the target size η
Result: A polynomial, the achieved precision K', the achieved size η'

$(P_{\mathbb{R}}, \gamma) :=$ floatApprox(K)
$(m_i)_{i\in[0,\gamma]}$ defined with Eq. (2.30)
B computed with Eq. (2.31).
$B' :=$ LLL-reduced B with respect to $|.|_{\mathrm{S}}$
$\frac{P_f^I}{f} :=$ Babai Reduction of $\frac{P_{\mathbb{R}}}{f}$ in B'.
K' computed with Eqs. (2.27) and (2.32)
$\eta' :=$ maximal size of the coefficients of P_f^I
return P_f^I, K', η'

In Algorithm 7, we informally sketch all the steps for finding our final approximation. The algorithm is ad-hoc and works by trial and error on the input parameters. For example, several cases can appear.

1. The final relative precision K' is too small compared to the target one. In this case, Algorithm 7 must be run with a higher input K in order to increase the degree γ.

2. The final size of the coefficients η' is larger than the target one. This happens when the lattice reduction was not successful enough to get a close vector in the lattice. Here too, taking a more significant input degree can help to get a finer approximation.

[3] Using BKZ was not relevant for the sizes manipulated. Besides, unlike for LLL, there is no function for BKZ that allows giving the Gram matrix as input.

2.3.3 The Sage tool

A SageMath 9.0 class (publicly available in [Bar+19b]) was implemented. It takes a function f, an interval I and a target (K, η) as inputs. It follows Algorithm 7 and generates a polynomial approximation of the function on I according to the previously described procedure. As the computation needs to perform many integrals, the tool calls Pari/GP instructions. The tool behaves very well on our applications (See Sections 2.4.3, 2.4.5 and 2.5.3) but as already stated, there are no general optimality statements on this procedure.

As an example, let us use the tool to approximate the function $x \mapsto \exp(-x)$ on $[1, 2]$ with targets $(K, \eta) = (20, 30)$.

```
sage: load('polynomial_approximation.sage')
....: a = 1  # Lower bound for I
....: b = 2  # Upper bound for I
....: f = "exp(-x)"
....: target_precision = 20  # target relative precision
....: eta = 25  # target size of coefficients
....: size_floats = 128  # precision of intermediate computations
....: galactics = GALACTICS_approximation(a, b, size_floats, target_precision, eta, f)
....: galactics.approximate()
```

```
.................................
Step 1 : look for a polynomial with float coefficients that gives more than 20 bits of
relative precision

iteration on the degrees :

degree 0 -> precision 0
degree 1 -> precision 2
degree 2 -> precision 5
degree 3 -> precision 8
degree 4 -> precision 12
degree 5 -> precision 17
degree 6 -> precision 21
Found a degree !

Conclusion: Approximating f(x) by a degree 6 polynomial in R ensures 21 bits of precision

The corresponding polynomial is: P =
0.00030639757680663614256554200837983692209*x^6 -
0.00463773699226475029998997068860002793 79*x^5 +
0.03374136991743839583427095617702879971 7*x^4 -
0.15596655425924862503523401481045802910*x^3 +
0.49116238390812574919951418501177974032*x^2 -
0.99591917917973812765784007336333646994*x +
0.99919276019463860932315882360313548431

.................................
Step 2 : Round the float coefficients to integer ones
Generating Lattice...
log(m[i]) =  [-25, -25, -26, -27, -29, -32, -36]
LLL reduction Done

Babai Rounding...

 ----------------------
 | Final security : 21  |
 | Max size coeffs : 25 |
 ----------------------
```

```
.................................

Exact polynomial:
P = 16763669 * 2 ^ -24
  + -33417471 * 2 ^ -25 * x ^ 1
  + 16480621 * 2 ^ -25 * x ^ 2
  + -10466641 * 2 ^ -26 * x ^ 3
  + 18114369 * 2 ^ -29 * x ^ 4
  + -4979525 * 2 ^ -30 * x ^ 5
  + 10527017 * 2 ^ -35 * x ^ 6
```

The final polynomial effectively provides 2^{-21} relative precision with $x \mapsto \exp(-x)$ on $[1, 2]$ and the sizes of its coefficients do not exceed 25.

2.4 Isochronous BLISS

The BLISS signature scheme introduced in 2013 in [Duc+13] was possibly the most efficient lattice-based signature scheme so far introduced at that time. It has been implemented in both software [Duc+13] and hardware [PDG14] and boasted performance numbers comparable to classical factoring and discrete-logarithm based schemes. BLISS can be seen as a ring-based optimization of the earlier lattice-based scheme of Lyubashevsky [Lyu12], sharing the same "Fiat–Shamir with aborts" structure [Lyu09].

> Please note that its parameters were chosen before the NIST standardization process and may now be considered not enough conservative compared to those of the NIST candidates.

2.4.1 BLISS signature scheme

Algorithm 8 — BLISS key generation

Result: Signing key sk, verification key pk

1 Generate two polynomials $\mathbf{f}$ and $\mathbf{g}$ uniformly at random in R_q with exactly $n\delta_1$ entries in $\{\pm 1\}$ and $n\delta_2$ entries in $\{\pm 2\}$
2 $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2) \coloneqq (\mathbf{f}, 2 \cdot \mathbf{g} + 1)$
3 **if** $N_\kappa(\boldsymbol{S}) \geq C^2 \cdot 5 \cdot (\lceil \delta_1 \cdot n \rceil + 4 \cdot \lceil \delta_2 \cdot n \rceil) \cdot \kappa$ **then**
4 | **restart** to step 2
5 **end**
6 $\mathbf{a}_q \coloneqq (2 \cdot \mathbf{g} + 1)/\mathbf{f} \mod q$ (**restart** if $\mathbf{f}$ is not invertible.)
7 $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2) \coloneqq (2 \cdot \mathbf{a}_q, q - 2) \mod 2q$
8 **return** $(pk, sk) \coloneqq (\mathbf{A}, \mathbf{S})$

One can give a simplified description of the scheme as follows: the public key pk is a (NTRU-like) ratio of the form $\mathbf{a}_q = \mathbf{s}_2/\mathbf{s}_1 \bmod q$, where the signing key sk is composed of polynomials $\mathbf{s}_1, \mathbf{s}_2 \in R_q$ (the cyclotomic ring is defined in Eq. (1.3)) are small and sparse. The integer n is a power of two. See Algorithm 8 for a description of the key generation. Its security is based on the ring variant of short integer solution (see Hard Problem 6). The parameters $\kappa, C, \delta_1, \delta_2, q, p$ are part of BLISS specifications, and we refer to [Duc+13] for the details on the function N_κ.

The BLISS signature procedure, described in Algorithm 9, works as follows. To sign a message m, one first generates commitment values $\mathbf{y}_1, \mathbf{y}_2 \in R_q$ with normally distributed coefficients (i.e. using the distribution $D_{\mathbb{Z},\sigma}$), and then computes a hash $\mathbf{c}$ of the message m together with $\zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ (where $\zeta(q-2) = 1 \bmod 2q$) using a cryptographic hash function, hash,

modeled as a random oracle taking values in the set of elements of R_q with exactly κ coefficients equal to 1 and the others to 0.

Algorithm 9 — BLISS Signature

Data: $m, pk = \mathbf{a}_1, sk = \mathbf{S}$
Result: Signature sig

1 $\mathbf{y}_1 \leftarrow D^n_{\mathbb{Z},\sigma}$, $\mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$
2 $\mathbf{u} \coloneqq \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$ /* $\zeta(q-2) = 1 \bmod 2q$ */
3 $\mathbf{c} \coloneqq \mathsf{hash}(\lfloor \mathbf{u} \rceil_{\mathtt{Mst}} \bmod p, m)$
4 choose a random bit b
5 $\mathbf{z}_1 \coloneqq \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6 $\mathbf{z}_2 \coloneqq \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7 **if** $\mathsf{RS}(\mathbf{z} \coloneqq (\mathbf{z}_1, \mathbf{z}_2), \mathbf{S}, \mathbf{c})$ **then**
8 | **restart** to step 1
9 **end**
10 $\mathbf{z}_2^\dagger \coloneqq (\lfloor \mathbf{u} \rceil_{\mathtt{Mst}} - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_{\mathtt{Mst}}) \bmod p$
11 **return** sig $\coloneqq (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

Algorithm 10 — Detailed function RS (Step 7 in Algorithm 9)

Data: $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2), \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2), \mathbf{c} \in R_q$
Result: a bit equal to 0 with probability
$1/\left(M \exp(-\|\mathbf{c} \cdot \mathbf{S}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle / \sigma^2)\right)$ and 1 otherwise

1 Compute

$$x_1 \in I_1 := \left[-\frac{\sigma^2}{\alpha^2}, 0\right] \text{ and } x_2 \in I_2 := \left[-\frac{2B_2\sigma}{\alpha}, \frac{2B_2\sigma}{\alpha}\right]$$

such that

$$x_1 \coloneqq \|\mathbf{c} \cdot \mathbf{S}\|^2 - \frac{\sigma^2}{\alpha^2} \text{ and } x_2 \coloneqq 2 \cdot \langle \mathbf{z}, \mathbf{c} \cdot \mathbf{S} \rangle$$

2 Generate a pair (u_1, u_2) of fixed-precision numbers uniformly at random in $[0, 1]^2$
3 Let $a \coloneqq 1$ if $u_1 \leq \exp(\frac{x_1}{2\sigma^2})$, and $a \coloneqq 0$ otherwise
4 Let $b \coloneqq 1$ if $\cosh(\frac{x_2}{2\sigma^2}) \cdot u_2 \leq 1$, and $b \coloneqq 0$ otherwise
5 **return** $a \wedge b$

The signature is the triple $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, with $\mathbf{z}_i \coloneqq \mathbf{y}_i + \mathbf{s}_i\mathbf{c}$, and a **rejection sampling** (RS) phase is performed to ensure that the distribution of $\mathbf{z}_i$ is independent of the private key (see Section 2.2.5 for the rejection sampling theory). The rejection rate is defined as $M := \exp\left(\frac{1}{2\alpha^2}\right)$ for α a parameter of the scheme. A detailed rejection sampling procedure is presented in Algorithm 10. It is written in a form that will be helpful for the isochronous transformation. Originally, the approach taken in BLISS paper relies on iterated Bernoulli trials with known constant probabilities. The variable time nature of these algorithms has led to multiple attacks. An alternate approach is to simply evaluate the values $\exp(\frac{x_1}{2\sigma^2})$ and $\cosh(\frac{x_2}{2\sigma^2})$ with sufficient precision, and compare them to uniform random values in $[0, 1]$. Here, the exp and cosh are separated to account for the different polynomial approximations. The intervals I_1 and I_2 are derived from BLISS construction.

The signature procedure also includes several optimizations on top of the above description. In particular, to improve the repetition rate, it targets a bimodal Gaussian distribution for the $\mathbf{z}_i$'s, so there is a random sign flip, in Step 4 of Algorithm 9, in their definition. In addition, to

Table 2.1: *Concrete parameters for BLISS.*

BLISS–	0	I	II	III	IV
n	256	512	512	512	512
q	7681	12289	12289	12289	12289
σ	100	215	107	250	271
δ_1, δ_2	.55 , .15	.3 , 0	.3 , 0	.42 , .03	.45, .06
`Tail` (bits dropped in $\lfloor\cdot\rceil_{\texttt{Mst}}$)	5	10	10	9	8
$p := \lfloor 2q/2^{\texttt{Tail}} \rfloor$	480	24	24	48	96
κ	12	23	23	30	39
α	0.5	1.0	0.5	0.7	0.55
B_2	2492	12872	11074	10206	9901
B_∞	530	2100	1563	1760	1613

reduce the signature size, the signature element $\mathbf{z}_2$ is actually transmitted in compressed form $\mathbf{z}_2^\dagger$, and accordingly. Thus, the element $\mathbf{u}$ is also compressed. The notation $\lfloor\cdot\rceil_{\texttt{Mst}}$ represents this compression: The `Tail` (parameter chosen in the specifications) least significant bits are dropped. The signature compression does not affect our technique at all, because it only involves non-sensitive parts of the signature generation algorithm (the input of the hash function and the returned signature itself).

The verification procedure is given in Algorithm 11 for completeness since it does not manipulate sensitive data. B_2 and B_∞ are parameters detailed in BLISS specifications.

Algorithm 11 — BLISS Verification

Data: m, sig, pk
if $\|(\mathbf{z}_1 \| 2^d \cdot \mathbf{z}_2^\dagger)\|_2 > B_2$ **then**
  return reject
end
if $\|(\mathbf{z}_1 \| 2^d \cdot \mathbf{z}_2^\dagger)\|_\infty > B_\infty$ **then**
  return reject
end
$t \coloneqq \mathsf{hash}(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_{\texttt{Mst}} + \mathbf{z}_2^\dagger \bmod p, m)$
if $t \neq \mathbf{c}$ **then**
  return reject
end
return accept

The original BLISS paper describes a family of concrete parameters for the signature scheme, stored in Table 2.1. The BLISS–I and BLISS–II parameter sets are optimized for speed and compactness respectively, and both target 128 bits of security. BLISS–III and BLISS–IV are claimed to offer 160 and 192 bits of security respectively. Finally, a BLISS–0 variant is also given as a toy example offering a relatively low security.

2.4.2 Isochrony

We propose here a provable isochronous implementation of the signature algorithm of BLISS, mainly relying on an alternate implementation of the rejection sampling step, carried out by computing a sufficiently precise polynomial approximation of the rejection probability using pure integer arithmetic.

The key generation of BLISS can also be protected using sorting networks. Nevertheless, the key generation is now obsolete in terms of complexity and performance compared to BLISS-B or more recent signature implementations (Dilithium, qTESLA). Thus, in this thesis, we focus on presenting the isochronous signature algorithm.

The goal of this section is to apply Lemma 6 for approximating :

- the $\exp(\cdot)$ in step 3 of Algorithm 10 (see Section 2.4.3);
- the $\cosh(\cdot)$ in step 4 of Algorithm 10 (see Section 2.4.4);
- the $\exp(\cdot)$ in step 1 of Algorithm 9 (see Section 2.4.5).

The final security and isochrony statements are provided in Section 2.4.6.

2.4.3 Polynomial approximation for the exponential

Choosing the approximation interval Recall that the input interval is $I_1 = \left[-\sigma^2/\alpha^2, 0\right]$. But, to anticipate further optimization, we approximate the polynomial in a slightly larger interval, but still finite,

$$I_3 := [-2 \cdot \ln(2) \cdot \sigma^2/\alpha^2, 0].$$

Choosing the factor K NIST suggested $\mathtt{Q}_\mathrm{s} = 2^{64}$ maximum signature queries for post-quantum standardization. During the rejection sampling, the call to exp() and cosh() is done $M < 2$ times per signatures on average, so the polynomial approximations used in the rejection sampling step can assume that the number of queries is $\mathtt{Q} \approx \mathtt{Q}_\mathrm{s} = 2^{64}$. We are willing to apply Lemma 6 with the original bit security of $\lambda = 256$ and $\mathtt{Q} = 2^{64}$. The target relative precision must be

$$\frac{1}{\sqrt{2\,(2\lambda - 1)\,\mathtt{Q}}} = 2^{-37}. \tag{2.33}$$

Thus, we look for a polynomial denoted $P_{\exp}^{I_3}$ that verifies

$$\forall x \in I_3, \quad \left| \frac{\exp(\frac{x}{2\sigma^2}) - P_{\exp}^{I_3}(x)}{\exp(\frac{x}{2\sigma^2})} \right| \leq 2^{-K} \tag{2.34}$$

with $K = 37$. In order to account for the slight loss of precision as part of the fixed point evaluation of the polynomials using Horner's rule, we need to take slightly more precise values: we verify that $K \geq 39$ suffices (one could use general error bounds for Horner such as [Hig02, Eq. (5.3)]. However, since our evaluations occur on small integer intervals, it is easy to check their precision by exhausting the intervals).

Using our approximation tool We use our polynomial approximation tool presented in Section 2.3 with BLISS-I parameters. The final polynomial has degree 10 and 45-bit coefficients.

```
sage: load('polynomial_approximation.sage')
....: galactics = GALACTICS_approximation(- 2*ln(2)*215^2/1^2, 0, 128, 45, 45,
                                          "exp(x / 2 / 215^2)")
....: galactics.approximate()
```

```
[...]
 ----------------------
 | Final security : 47  |
 | Max size coeffs : 45 |
 ----------------------
................................

Exact polynomial:
P = 1 * 2 ^ 0
  + 6235378607933 * 2 ^ -59 * x ^ 1
  + 17680552620763 * 2 ^ -78 * x ^ 2
  + 33422396137565 * 2 ^ -97 * x ^ 3
  + 5923120905557 * 2 ^ -113 * x ^ 4
  + 26872204609341 * 2 ^ -134 * x ^ 5
  + 25398480056119 * 2 ^ -153 * x ^ 6
  + 20570037805739 * 2 ^ -172 * x ^ 7
  + 3629577032917 * 2 ^ -189 * x ^ 8
  + 35066854785757 * 2 ^ -212 * x ^ 9
  + 14707071337199 * 2 ^ -231 * x ^ 10
```

Experimental verification of Eq. (2.24) To comply with Lemma 6, we must check that $\forall x \in I_3, \left| \frac{\exp\left(\frac{x}{2\sigma^2}\right) - P^{I_3}_{\exp}(x)}{1-\exp\left(\frac{x}{2\sigma^2}\right)} \right| \leq 2^{-40}$.

Perspective 1. *The condition of Eq. (2.24) has been added in a recent modification of this thesis. An interesting line of research would be to take Eq. (2.24) directly into account in the Galactics polynomial approximation to obtain both bounds.*

We experimentally check that Eq. (2.24) is verified with the outputted polynomial. First, we notice that such a bound exists by studying the variations of the left member. More precisely, this member decreases when $x \to 0$, this can be seen with computations on $x = -2^{-i}$ for large i in high precision. The limit when $x \to 0$ is also a finite very small number. Then, we compute the maximum by exhausting I_3. Indeed, we recall that for BLISS, the input x is an integer in I_3. We finally have

$$\forall x \in I_3, \left| \frac{\exp\left(\frac{x}{2\sigma^2}\right) - P^{I_3}_{\exp}(x)}{1-\exp\left(\frac{x}{2\sigma^2}\right)} \right| \leq 2^{-44}, \tag{2.35}$$

which is enough to validate the condition of Eq. (2.24).

2.4.4 Polynomial approximation of the hyperbolic cosine

The input interval for the hyperbolic cosine is large, namely

$$I_2 = \left[-\frac{2B_2\sigma}{\alpha}, \frac{2B_2\sigma}{\alpha} \right] \approx [-5534960, 5534960]$$

for BLISS-I. Due to the parity of the hyperbolic cosine, the study on I_2 can be reduced to $I_2 = \left[0, \frac{2B_2\sigma}{\alpha}\right] \approx [0, 5534960]$ for BLISS-I. The target approximation factor is the same as for the exponential, namely $K \geq 39$.

First Approach After one day of computations, the following instructions succeeded to output a valid polynomial with 48 coefficients of 110 bits.

```
sage: load('polynomial_approximation.sage')
....: galactics = GALACTICS_approximation(0, 5534960, 512, 65, 100,
                                          "(exp(x / 2 / 215^2) + exp(- x / 2 / 215^2) / 2")
....: galactics.approximate()
```

```
[...]

 ------------------------
 | Final security : 55   |
 | Max size coeffs : 109 |
 ------------------------
.................................

Exact polynomial:
P = 1 * 2 ^ 0
  + 579356348280174377 * 2 ^ -93 * x ^ 2
  + 776355318672508413 * 2 ^ -130 * x ^ 4
  + 915092737617992344722936430591 * 2 ^ -208 * x ^ 6
  + 477972884153784129 * 2 ^ -206 * x ^ 8
  + 683198250418101963 * 2 ^ -246 * x ^ 10
  + 732080158605593691080137637887 * 2 ^ -326 * x ^ 12
  + 941246584139471315 * 2 ^ -327 * x ^ 14
  + 1009039488665294857 * 2 ^ -368 * x ^ 16
  + 848404038966405199 * 2 ^ -409 * x ^ 18
  + 574427366621078729 * 2 ^ -450 * x ^ 20
  + 639792480494866645 * 2 ^ -492 * x ^ 22
  + 596411567633024933 * 2 ^ -534 * x ^ 24
  + 1038265388175552617133165772799 * 2 ^ -617 * x ^ 26
  + 642737561059887937 * 2 ^ -619 * x ^ 28
  + 835957399885546368823128489983 * 2 ^ -702 * x ^ 30
  + 788833826198999147 * 2 ^ -705 * x ^ 32
  + 180800091273276529 * 2 ^ -746 * x ^ 34
  + 296114798686759457 * 2 ^ -790 * x ^ 36
  + 940728566765727412999436632063 * 2 ^ -875 * x ^ 38
  + 150149332642775351 * 2 ^ -876 * x ^ 40
  + 57245075745611325969266638847 9 * 2 ^ -962 * x ^ 42
  + 748157035942843819 * 2 ^ -965 * x ^ 44
  + -2318931663498473299 * 2 ^ -1010 * x ^ 46
  + 11469920785630361661 * 2 ^ -1054 * x ^ 48
  + -720704975489956588178017804615 67 * 2 ^ -1139 * x ^ 50
  + 45346083844645413249 * 2 ^ -1141 * x ^ 52
  + -857389151161771913179 * 2 ^ -1188 * x ^ 54
  + 112389859384283396677 * 2 ^ -1228 * x ^ 56
  + -414355076723513568697 * 2 ^ -1273 * x ^ 58
  + 337060337629183540215 * 2 ^ -1316 * x ^ 60
  + -265830663567489890942195238699007 * 2 ^ -1399 * x ^ 62
  + 336319019176325683112175007170559 * 2 ^ -1443 * x ^ 64
  + -170475232994946739825 * 2 ^ -1446 * x ^ 66
  + 167183977538710670933 * 2 ^ -1490 * x ^ 68
  + -71970509923746855587 * 2 ^ -1533 * x ^ 70
  + 216967001776759787417 * 2 ^ -1579 * x ^ 72
  + -35646442335471927163 * 2 ^ -1621 * x ^ 74
  + 8129818103264009 1129 * 2 ^ -1667 * x ^ 76
  + -159830707981302343261 * 2 ^ -1713 * x ^ 78
  + 14767767207256590536810248011775 9 * 2 ^ -1798 * x ^ 80
  + -2386357485300151311 7 * 2 ^ -1801 * x ^ 82
  + 113128579100766722931 * 2 ^ -1849 * x ^ 84
  + -109674964769571482255 * 2 ^ -1895 * x ^ 86
  + 21170189159645535701 * 2 ^ -1939 * x ^ 88
  + -3128582308920305019 * 2 ^ -1983 * x ^ 90
  + 85067057155831645461 * 2 ^ -2035 * x ^ 92
  + -11559169938839276831 * 2 ^ -2080 * x ^ 94
  + 96586818007198473957 * 2 ^ -2132 * x ^ 96
```

This approximation is convenient for the masking countermeasure (we refer to the masking chapter in Section 3.4.1). However, the degree and size of the coefficients are large. In isochrony, an optimization using shifts can be done.

Second approach We use the following lemma for a more convenient approximation.

Lemma 8. *Let* $c \coloneqq 2\sigma^2 \ln(2)$. *Assume that* $P_{\exp}^{I_3}$ *is a precision* K *approximation of* $\exp\left(\frac{\cdot}{2\sigma^2}\right)$ *on* I_3. *Let us define the following function.*

$$\begin{array}{rccl} t_c: & I_2 & \to & I_3 \\ & x & \mapsto & x - \left\lfloor \frac{x}{c} \right\rfloor c. \end{array} \tag{2.36}$$

The following polynomial

$$P_{\cosh}^{I_2}(x) \coloneqq \frac{\overbrace{2^{\left\lfloor \frac{x}{c} \right\rfloor} \cdot P_{\exp}^{I_3}\left(t_c\left(x\right)\right)}^{=\exp\left(\frac{x}{2\sigma^2}\right)} + 2^{-\left\lfloor \frac{x}{c} \right\rfloor} \cdot \exp\left(\frac{c}{2\sigma^2}\right) \cdot P_{\exp}^{I_3}\left(-t_c\left(x\right) - c\right)}{2} \tag{2.37}$$

is a precision $K-1$ *approximation of* $\cosh\left(\frac{\cdot}{2\sigma^2}\right)$ *on* I_2.

Proof: By definition, $t_c(x)$ belongs in I_3, thus

$$\forall x \in I_2 \quad \frac{|P_{\exp}^{I_3}(t_c(x)) - \exp\left(\frac{t_c(x)}{2\sigma^2}\right)|}{|\exp\left(\frac{t_c(x)}{2\sigma^2}\right)|} \leq 2^{-K}$$

Then, let us compute the relative error for $P_{\cosh}^{I_2}$. For $x \in I_2$,

$$\begin{aligned}
\frac{|P_{\cosh}^{I_2}(x) - \cosh\left(\frac{x}{2\sigma^2}\right)|}{|\cosh\left(\frac{x}{2\sigma^2}\right)|} &= \left| \frac{2^{\left\lfloor \frac{x}{c} \right\rfloor} \cdot P_{\exp}^{I_3}(t_c(x)) + 2^{-\left\lfloor \frac{x}{c} \right\rfloor} \cdot \exp\left(\frac{c}{2\sigma^2}\right) \cdot P_{\exp}^{I_3}(-t_c(x) - c) - \exp\left(\frac{x}{2\sigma^2}\right) - \exp\left(\frac{-x}{2\sigma^2}\right)}{\exp\left(\frac{x}{2\sigma^2}\right) + \exp\left(\frac{-x}{2\sigma^2}\right)} \right| \\
&\leq \left| \frac{2^{\left\lfloor \frac{x}{c} \right\rfloor} \cdot P_{\exp}^{I_3}(t_c(x)) - \exp\left(\frac{x}{2\sigma^2}\right)}{\exp\left(\frac{x}{2\sigma^2}\right) + \exp\left(\frac{-x}{2\sigma^2}\right)} \right| + \left| \frac{2^{-\left\lfloor \frac{x}{c} \right\rfloor} \cdot \exp\left(\frac{c}{2\sigma^2}\right) \cdot P_{\exp}^{I_3}(-t_c(x) - c)) - \exp\left(\frac{-x}{2\sigma^2}\right)}{\exp\left(\frac{x}{2\sigma^2}\right) + \exp\left(\frac{-x}{2\sigma^2}\right)} \right| \\
&\leq \left| \frac{2^{\left\lfloor \frac{x}{c} \right\rfloor} \cdot P_{\exp}^{I_3}(t_c(x)) - \exp\left(\frac{x}{2\sigma^2}\right)}{\exp\left(\frac{x}{2\sigma^2}\right)} \right| + \left| \frac{2^{-\left\lfloor \frac{x}{c} \right\rfloor} \cdot \exp\left(\frac{c}{2\sigma^2}\right) \cdot P_{\exp}^{I_3}(-t_c(x) - c) - \exp\left(\frac{-x}{2\sigma^2}\right)}{\exp\left(\frac{-x}{2\sigma^2}\right)} \right| \\
&= \left| \frac{P_{\exp}^{I_3}(t_c(x)) - \exp\left(\frac{t_c(x)}{2\sigma^2}\right)}{\exp\left(\frac{t_c(x)}{2\sigma^2}\right)} \right| + \left| \frac{2^{-\left\lfloor \frac{x}{c} \right\rfloor} \exp\left(\frac{c}{2\sigma^2}\right) \cdot P_{\exp}^{I_3}(-t_c(x) - c) - \exp\left(\frac{-t_c(x)}{2\sigma^2}\right) 2^{-\left\lfloor \frac{x}{c} \right\rfloor}}{2^{-\left\lfloor \frac{x}{c} \right\rfloor} \exp\left(\frac{-t_c(x)}{2\sigma^2}\right) \cdot \exp\left(\frac{c}{2\sigma^2}\right)} \right| \\
&\leq \left| \frac{P_{\exp}^{I_3}(t_c(x)) - \exp\left(\frac{t_c(x)}{2\sigma^2}\right)}{\exp\left(\frac{t_c(x)}{2\sigma^2}\right)} \right| + \left| \frac{P_{\exp}^{I_3}(\overbrace{-t_c(x) - c}^{\in I_3}) - \exp\left(\frac{-t_c(x) - c}{2\sigma^2}\right)}{\exp\left(\frac{-t_c(x) - c}{2\sigma^2}\right)} \right| \\
&\leq 2^{-K} + 2^{-K} = 2^{-K+1}.
\end{aligned}$$

□

Note that Eq. (2.37) can be computed exactly in a fast and isochronous way. Indeed, the factor $2^{\left\lfloor \frac{x}{c} \right\rfloor}$ consists in a bit shift. Namely, it consists of at most 86 shifts for BLISS-I (because $\left\lfloor \frac{x}{c} \right\rfloor \leq \frac{B_2}{\alpha \ln(2) \sigma}$).

Note that Lemma 6 cannot be applied with both approaches because the requirement of $\forall x \in I, 0 \leq f(x) \leq 1$ is not verified. We introduce one last lemma to handle this issue.

Lemma 9. *Assume that* $P^{I_2}_{\cosh}(x)$ *is a precision* K *approximation of* $\cosh\left(\frac{\cdot}{2\sigma^2}\right)$ *on* I_2. *Then,* $\frac{1}{P^{I_2}_{\cosh}(x)}$ *is also a precision* K *approximation of* $\frac{1}{\cosh\left(\frac{\cdot}{2\sigma^2}\right)}$ *on* I_2.

Proof: For $x \in I_2$, we have

$$\left|\frac{\frac{1}{2P^{I_2}_{\cosh}(x)} - \frac{1}{2\cosh\left(\frac{x}{2\sigma^2}\right)}}{\frac{1}{2\cosh\left(\frac{x}{2\sigma^2}\right)}}\right| = \left|\frac{P^{I_2}_{\cosh}(x) - \cosh\left(\frac{x}{2\sigma^2}\right)}{P^{I_2}_{\cosh}(x)}\right|.$$

Next, we want to us the following result,

$$\forall p, q \in \mathbb{R} \quad \left|\frac{p-q}{p}\right| \leq \left|\frac{p-q}{q}\right| + \left|\frac{p-q}{q}\right|^2. \tag{2.38}$$

Let us first prove Eq. (2.38). We have $\left|\frac{p-q}{p}\right| = \left|\frac{p-q}{q}\right| \cdot \left|\frac{q}{p}\right|$. If $|q| \leq |p|$, the result is direct. If $|q| \geq |p|$, we have $\left|\frac{q}{p}\right| \leq \left|\frac{p}{q}\right| \leq 1 + \left|\frac{p-q}{q}\right|$.

The application of Eq. (2.38) to $p = P^{I_2}_{\cosh}(x)$ and $q = \cosh\left(\frac{x}{2\sigma^2}\right)$ allows to conclude with:

$$\left|\frac{\frac{1}{2P^{I_2}_{\cosh}(x)} - \frac{1}{2\cosh\left(\frac{x}{2\sigma^2}\right)}}{\frac{1}{2\cosh\left(\frac{x}{2\sigma^2}\right)}}\right| \lesssim \left|\frac{P^{I_2}_{\cosh}(x) - \cosh\left(\frac{x}{2\sigma^2}\right)}{\cosh\left(\frac{x}{2\sigma^2}\right)}\right| \leq 2^{-K}.$$

□

Experimental verification of Eq. (2.24) An experimental verification of the condition in Eq. (2.24) can be performed similarly to Section 2.4.3. For the first approach, by exhausting I_2, we get

$$\forall x \in I_2, \left|\frac{\cosh\left(\frac{x}{2\sigma^2}\right) - P^{I_2}_{\cosh}(x)}{1 - \cosh\left(\frac{x}{2\sigma^2}\right)}\right| \leq 2^{-56} \leq 2^{-39}.$$

For the second approach, the bound can be derived from Eq. (2.35) with a proof similar to the one of Lemma 9 (except that there is an $1 - \exp$ in the denominator). Thus, we get

$$\forall x \in I_2, \left|\frac{\cosh\left(\frac{x}{2\sigma^2}\right) - P^{I_2}_{\cosh}(x)}{1 - \cosh\left(\frac{x}{2\sigma^2}\right)}\right| \leq 2^{-44+1} \leq 2^{-39}.$$

2.4.5 Isochronous Gaussian generation for BLISS

The Gaussian sampling step is key to obtaining a fast implementation of BLISS, as it represents half or more of the computation time of signature generation: for each signature, one needs to generate 1024 samples of the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ (possibly several times over, in case a rejection occurs), and the standard deviation is relatively large ($\sigma = 215$ for BLISS–I). This step has also been specifically targeted by cache timing attacks such as [Bru+16]. In the following, we review the different possible techniques.

Direct CDT One can aim at drawing from $|D_{\mathbb{Z},\sigma}|$ (as defined in Section 2.2.4) with a CDT and draw a uniformly random sign. Since the standard deviation is a fixed parameter and the center is 0, one can consider the CDT sampling presented in Section 2.2.4. Here, for ensuring 128-bit precision on the table elements, we need to make sure that the number of elements of the table, denoted w, is defined such that $\exp(-w^2/(2\sigma^2)) < 2^{-\lambda}$. Thus, the table should contain $w = \sigma\sqrt{2\lambda \log 2} \approx 2730$ entries for BLISS–I, we are looking at 22 kB's worth of memory access for every generated sample. The resulting implementation is highly inefficient.

Using Convolution methods One might want to apply convolution techniques like [MW17] (presented in Section 2.2.4). For the parameters of BLISS–I, one can check that the optimal choice is to let $\sigma_0^2 = \frac{\sigma^2}{(9^2+7^2)(3^2+2^2)}$. One can then generate a sample x statistically close to $|D_{\mathbb{Z},\sigma}|$ from 4 samples $x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}$ from $|D_{\mathbb{Z},\sigma_0}|$, as $x = 9x_0 + 7x_1$, where $x_i = 3x_{i,0} + 2x_{i,1}$. Since $\sigma_0 \approx 4.99$ is much smaller than σ, using a CDT approach for the base sampler is more reasonable: the CDT table now stores 63 entries. Generating a sample requires reading through the table 4 times, for a total of 2 kB of memory access and 128 bits of randomness per sample. It turns out, however, that the performance of the resulting implementation in our setting is still somewhat underwhelming.

A strategy for Gaussian sampling with large standard deviations Finally, yet another strategy is to generate a discrete Gaussian of a fixed small standard deviation, use it to construct a distribution that looks like $D_{\mathbb{Z}^+,\sigma}$, and use rejection sampling to correct the discrepancy. This is actually the approach taken in the original BLISS paper [Duc+13]. Concretely, the sampling is done using the distribution $D_{\mathbb{Z}^+,\sigma_2}$ where $\sigma_2 = \sigma/k$ for a given parameter k. The sampling is described in Algorithm 12.

Algorithm 12 — BLISS-like Gaussian Sampling

Params: k, σ
Result: A sample z following $D_{\mathbb{Z}^+,\sigma}$

1 $x \leftarrow D_{\mathbb{Z}^+,\sigma/k}$
2 $y \xleftarrow{\$} \{0, \ldots, k-1\}$
3 $z := k \cdot x + y$
4 **Goto** 1 with probability $\exp\left(-y(y+2kx)/(2\sigma^2)\right)$
5 **if** $z = 0$ **then**
6 | **Goto** 1 with probability 1/2
7 **end**
8 $b \xleftarrow{\$} \{0, 1\}$
9 **return** $(-1)^b \cdot z$

The idea of Steps 1 to 4 is that $z = kx + y$ looks "somewhat like" a sample from $D_{\mathbb{Z}^+,\sigma}$, and rejecting z except with probability $\exp\left(-y(y+2kx)/(2\sigma^2)\right)$ yields a value that actually follows $D_{\mathbb{Z}^+,\sigma}$. Let us detail why the latter is true. We apply Lemma 5 from Section 2.2.5. Here, we are in the particular case where g_v is not a family of distributions but a single distribution without being indexed by v. The distribution function g is

$$g(z) = g(kx + y) = \frac{\rho_{\sigma_2}(x)}{k\rho_{\sigma_2}(\mathbb{Z}^+)}$$

because, once x is drawn, by Euclidian division properties, z is uniquely written as $kx + y$ for $y \in \{0, \ldots, k-1\}$. And the desired output distribution function f is

$$f(z) = f(kx + y) = \frac{\rho_{k\sigma_2}(kx + y)}{\rho_{k\sigma_2}(\mathbb{Z}^+)}.$$

Now the ratio is proportional (written $\propto$) to

$$\begin{aligned} f(z)/g(z) &\propto \frac{\rho_{k\sigma_2}(kx+y)}{\rho_{\sigma_2}(x)} = \exp\left(-\frac{(kx+y)^2}{2k^2\sigma_2^2} + \frac{x^2}{2\sigma_2^2}\right) \\ &\propto \exp\left(-\frac{k^2x^2+y^2+2kxy}{2\sigma^2} + \frac{k^2x^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{y(y+2kx)}{2\sigma^2}\right). \end{aligned}$$

In addition, one can compute the constant average repetition rate as $M = \left\| \frac{f}{g} \right\|_\infty$.

The idea of Steps 5 to 9 is to draw a random sign. More precisely, at Step 5, z follows $D_{\mathbb{Z}^+,\sigma}$, which is not exactly $|D_{\mathbb{Z},\sigma}|$ (see Remark 1). We thus need to divide the probability of 0 by two; it is performed with a simple rejection sampling step.

Making it isochronous. One can think that because of the probabilistic property of Step 4 in Algorithm 12, this cannot be isochronous. However, thanks to the statistical indistinguishability provided by Lemma 5, the number of iterations follows a *geometric distribution with parameter* M and thus is independent from any secrets.

One ingredient to add is a base sampler for the distribution $|D_{\mathbb{Z},\sigma_2}|$, since the one in the original BLISS paper does not lend itself to a convenient isochronous implementation. Fortunately, choosing

$$k = 256,$$

makes the standard deviation $\sigma_2 \approx 0.80$ very small, and hence a CDT approach only requires 10 table entries for achieving 128-bit absolute precision.

The second ingredient is the rejection sampling. As observed in the BLISS paper, this rejection sampling step (Step 4 of Algorithm 12) has the same form as the one used for the overall signing algorithm (Step 7 in Algorithm 9, detailed in Algorithm 10). The isochronous implementation of that step has been described in Algorithm 10, and we can simply reuse that work to obtain our Gaussian sampling. In particular, we can derive a simple rejection sampling procedure (see Procedure 1) with a Bernoulli distribution.

Procedure 1 — Detailed Rejection Sampling of BLISS Gaussian Sampler (Step 4 in Algorithm 12)

1 $x_3 \coloneqq -y(y + 2kx)$
2 Generate u a fixed-precision number uniformly at random in $[0, 1]$
3 **return** 1 if $u \leq \exp(\frac{x_3}{2\sigma^2})$, and 0 otherwise

Here too, we use GALACTICS to generate a polynomial approximation of $\exp\left(\frac{\cdot}{2\sigma^2}\right)$. There are two slight differences with the previous approximation.

1. The relative target precision must be higher because the number of queries to the distribution is also higher. More precisely, the Bernoulli distribution with exponential parameter is called once per attempt at generating a Gaussian sample. The number of attempts is small (≤ 2 on average) due to rejection in Gaussian sampling. However, the Gaussian sampling is repeated $2n$ times to generate all the coefficients of $\mathbf{y}_1, \mathbf{y}_2$, and M times overall, where M is the repetition rate of the signature scheme. Therefore, the expected number of calls to the Bernoulli distribution as part of Gaussian sampling when generating $\mathtt{Q}_\mathrm{s}$ signatures, is bounded as $\mathtt{Q} \leq 2M \cdot 2n \cdot \mathtt{Q}_\mathrm{s} \leq 2^{78}$ for BLISS–I. This implies that, in order to apply Lemma 6, the relative target precision must be computed for $\lambda = 256$ and $\mathtt{Q} = 2^{78}$,

$$\frac{1}{\sqrt{2 \cdot (2 \cdot \lambda - 1) \cdot \mathtt{Q}}} = 2^{-44}. \tag{2.39}$$

2. We use the Euclidean division optimization of Eq. (2.36), which is used for the hyperbolic cosine approximation. In that way, the approximation interval is $I_3 = [-2\sigma^2 \ln(2), 0]$ and a fixed number of shifts are used to compute $2^{\left\lfloor \frac{x_3}{2\sigma^2 \ln(2)} \right\rfloor}$.

Here is the new polynomial approximation.

```
sage: load('polynomial_approximation.sage')
....: galactics = GALACTICS_approximation( - 2 * 215^2 * ln(2), 0, 128, 48, 45,
                                           "exp(x / 2 / 215^2)")
....: galactics.approximate()
```

```
[...]
  ----------------------
 | Final security : 50  |
 | Max size coeffs : 45 |
  ----------------------
.................................

Exact polynomial:
P = 1 * 2 ^ 0
  + 24941514431733 * 2 ^ -61 * x ^ 1
  + 4420138155217 * 2 ^ -76 * x ^ 2
  + 33422396152215 * 2 ^ -97 * x ^ 3
  + 11846242059507 * 2 ^ -114 * x ^ 4
  + 26872223790743 * 2 ^ -134 * x ^ 5
  + 12699467954197 * 2 ^ -152 * x ^ 6
  + 20576945247259 * 2 ^ -172 * x ^ 7
  + 29170523303177 * 2 ^ -192 * x ^ 8
  + 2292170444093 * 2 ^ -208 * x ^ 9
  + 20154220626817 * 2 ^ -231 * x ^ 10
  + 15924804363279 * 2 ^ -251 * x ^ 11
```

To check Eq. (2.24), similarly to Section 2.4.3, we compute

$$\forall x \in I_3, \left| \frac{\exp\left(\frac{x}{2\sigma^2}\right) - P_{\exp}^{I_3}(x)}{1 - \exp\left(\frac{x}{2\sigma^2}\right)} \right| \leq 2^{-55}.$$

Finally, in practice, this yields a Gaussian sampling of very reasonable efficiency, whose cost is dominated by the cost of the rejection sampling step, and of the generation of the uniform randomness. This is the approach we choose for our work.

The presented Gaussian sampler can be seen as a variant of FACCT [ZSS18]. There are multiple differences between our works, however: in particular, FACCT relies on floating-point arithmetic, which we specifically seek to avoid, and uses off-the-shelf software to obtain a double-precision floating-point polynomial approximation of the function exp.

Perspective 2. *The authors of the qTesla*[4] *second round NIST submission [Bin+19] proposed an ingenious approach to improve isochronous CDT-based discrete Gaussian sampling. In practice, one needs to generate many samples from the discrete Gaussian distribution in each signature (one for each coefficient of the* $\mathbf{y}_i$ *polynomials). The idea is to batch all of the searches through the CDT table corresponding to those samples. This can be done isochronously by applying an oblivious sorting algorithm (e.g. network sorting) to the concatenation of the CDT with the list of uniform random samples. This can be used in conjunction with the convolution technique of [PDG14; MW17] in order to reduce the total size of the table to be sorted (which is the sum of the CDT size and of the desired number of samples). A detailed investigation of this question is left as interesting perspective.*

2.4.6 Security and isochrony statement

We denote by I-BLISS the modified version of the original BLISS (as described in [Duc+13]) with all the modifications presented in the last sections. In other words, I-BLISS is a BLISS version where

[4] While qTesla uses uniform randomness during signature generation, it does use discrete Gaussians for key generation.

1. as explained in Section 2.4.5: in step 1, the Algorithm 12 is called with
   - a CDT sampling with 128-bit absolute precision instead of $D_{\mathbb{Z}^+,\sigma_2}$,
   - the polynomial in Section 2.4.5 instead of $\exp\left(\frac{\cdot}{2\sigma^2}\right)$;
2. as explained in Section 2.4.3: in step 7, the Algorithm 10 is called with
   - $P^{I_1}_{\exp}$ instead of $\exp\left(\frac{\cdot}{2\sigma^2}\right)$,
   - $P^{I_2}_{\cosh}$ instead of $\cosh\left(\frac{\cdot}{2\sigma^2}\right)$ (either with or without the optimization).

Theorem 1. *Let us assume that the original BLISS cryptosystem achieves λ-bit security by making at most $\mathtt{Q}_\mathrm{s} = 2^{64}$ signature queries. Then, the bit-security of I-BLISS achieves $(\lambda - 3)$-bit security.*

Proof: The proof consists in calling three times Lemma 6.[5] More precisely, let us introduce intermediate cases.

1. (Ideal) an unmodified version of BLISS signature is called.
2. (Inter$_1$) Only the exponential is approximated.
3. (Inter$_2$) Only the hyperbolic cosine is approximated.
4. (Real) The exponential in Algorithm 12 is also approximated.

The Ideal case is assumed to achieve λ-bit security. We first apply Lemma 6 to our final polynomial approximation for assessing the bit security of the Inter$_1$ case. Eqs. (2.34) and (2.35) show that Lemma 6 hypothesis are verified. In Section 2.4.3, K is defined to validate Eq. (2.33) and it is chosen as $K = 39$, we thus get that $\lambda_{\text{Inter}_1} \geq \lambda - 1$. Secondly, we apply Lemma 6 for assessing the bit security of the Inter$_2$ case. Here too, K is chosen as $K = 39$. Thanks to Lemma 9 and the experimental verifications that follow, we get that $\lambda_{\text{Inter}_2} \geq \lambda_{\text{Inter}_1} - 1 \geq \lambda - 2$. We finally apply Lemma 6 one last time for assessing the bit security of the Real case where the exponential in Algorithm 12 is approximated. Here too, K is larger due to larger number of queries, $K = 47$. Similarly, we get $\lambda_{\text{Real}} \geq \lambda_{\text{Inter}_2} - 1 \geq \lambda - 3$.

□

Proposition 3. *For any execution of our implementation of the signature generation $(\sigma, \mathscr{L}_{\mathsf{Sign}}) \leftarrow \mathsf{ExecObs}(\mathsf{Sign}, \mu, pk, sk)$, the leakage $\mathscr{L}_{\mathsf{Sign}}$ can be perfectly publicly simulated from the number of executions of the main loop. The latter, by Lemma 5, follows exactly a geometric law of parameter M, independent from any secret.*

Indeed, we have made sure that each step of the algorithm, except for the rejection sampling, is devoid of secret-dependent branches or memory accesses. As a result, the sequence of visited program points and memory accesses from each step is perfectly publicly simulatable, and the overall leakage $\mathscr{L}_{Sign}$ is obtained by repeating those simulations a number of times that is equal to the number of executions of the main loop. From these two results, together with the security of the rejection sampling, and the EUF-CMA security of BLISS, we can deduce that our implementation achieves I-EUF-CMA security.

2.4.7 Performance and implementation

Mehdi Tibouchi implemented our isochronous BLISS for the SUPERCOP toolkit for measuring cryptographic software performance [Bo16]. Note that the implementation is not completely up

[5] We thank Damien Stehlé for noting that Lemma 4 cannot be used directly.

Table 2.2: *Performance results and comparison (kcycles).*

	Median	Isochronous?
Dilithium (`ref`)	515	Yes
Dilithium (`avx2`)	332	Yes
qTesla-I (`ref`)	418	Yes
Original BLISS	194	No
Our implementation	220	Yes

to date with this thesis description (e.g. the polynomials in the implementation are not the latest ones).

We also provide a comparison to Ducas and Lepoint's original, variable-time implementation of BLISS on the same platform [DL13].

The Dilithium performance numbers are, for the fastest parameter set available in SUPERCOP, namely the `dilithium2` implementation, which corresponds to "medium" security parameters in [Duc+18] (no implementation is provided for the "weak" parameters). Timings both for the portable C (`ref`) and AVX2 platform-specific (`avx2`) implementations are given in Table 2.2. For qTesla, we also use the fastest available implementation (`qtesla1`, only in portable C[6]), which corresponds to essentially the same lattice security level as BLISS–I. The performance in Table 2.2 is consistent with the one publicly provided[7] with equivalent CPUs.

As we can see in the table, the performance level achieved here is similar to the original variable-time BLISS implementation, and the serious timing attack vulnerabilities exposed in multiple papers are prevented.

Legitimacy of the comparison In addition, our implementation is faster than qTesla-I and the portable C implementation of Dilithium. It even outperforms the AVX2 implementation of Dilithium by a significant margin, while providing stronger isochrony guarantees (since the Dilithium ring-valued hash function presents a mild timing leakage that causes the isochrony security to rely on non-standard assumptions). Admittedly, the Dilithium parameters were derived using a more conservative methodology for assessing the cost of lattice attacks and hence probably achieve a significantly higher level of security against them. Besides, its reliance on MLWE implies larger parameter sizes. Nevertheless, according to Wunderer's recent reevaluation [Wun19] of what is likely the strongest attack against BLISS (namely the Howgrave-Graham hybrid attack), it is reasonable to think that BLISS–I does reach its stated security level of around 128 bits.

2.5 A generic isochronous sampler for Falcon

In this section, we propose a generic isochronous Gaussian sampler over the integers based on the ideas of the previous sections. It is more precisely the 'small standard deviation' counterpart of our BLISS isochronous Gaussian sampling. This work [How+20] has been done with James Howe, Thomas Prest and Thomas Ricosset.

[6]The "heuristic" qTesla-I parameters were recently removed from the qTesla submission documents, and the remaining "provable" parameters are significantly less efficient. Since our goal is to compare it to comparable *fast* schemes, qTesla-I appears to be the most suitable parameter choice.

[7]`https://bench.cr.yp.to`

This sampler was originally designed for the Falcon signature scheme [Pre+19], but we figured that it is actually generic and can be applied in other settings. It has all the properties which are expected of a sampler for widespread deployment. It is simple and modular, making analysis and subsequent improvements easy. It is efficient and portable, making it amenable to a variety of scenarios.

The main assumption of our setting is to consider that all the standard deviations are bounded and that the center is in $[0, 1]$. In other words, denoting the upper bound and lower bound on the standard deviation as $\sigma_{\max} > \sigma_{\min} > 0$, we present an algorithm that samples the distribution $D_{\mathbb{Z},\sigma,\mu}$ for any $\mu \in [0, 1]$ and $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$.

2.5.1 The Gaussian sampler

Our sampling algorithm is called SamplerZ and is described in Algorithm 13. It is based on the general idea of constructing a distribution that looks somewhat like the desired Gaussian and use rejection sampling to correct the discrepancy.

Algorithm 13 — Gaussian Sampling (SamplerZ)

Data: $\mu \in [0, 1]$, $\sigma \leq \sigma_{\max}$
Result: A sample z following $D_{\mathbb{Z},\sigma,\mu}$

1 **while** *True* **do**
2 $\quad z_0 \leftarrow D_{\mathbb{Z}^+,\sigma_{\max}}$
3 $\quad b \xleftarrow{\$} \{0, 1\}$
4 $\quad z := (2b - 1) \cdot z_0 + b$
5 $\quad x := \frac{z_0^2}{2\sigma_{\max}^2} - \frac{(z-\mu)^2}{2\sigma^2}$
6 $\quad$ **if** $\mathsf{RS}(\sigma, x)$ **then**
7 $\quad\quad$ **return** z
8 $\quad$ **end**
9 **end**

The first step consists in sampling an element with a fixed half Gaussian distribution. The obtained z_0 sample is then transformed into $z := (2b-1) \cdot z_0 + b$ where b is a bit drawn uniformly in $\{0, 1\}$. Let us denote by $BG_{\sigma_{\max}}$ the distribution of z. The distribution of z is a discrete bimodal half-Gaussian of centers 0 and 1. More formally, for any $z \in \mathbb{Z}$,

$$BG_{\sigma_{\max}}(z) := \frac{1}{2} \begin{cases} D_{\mathbb{Z}^+,\sigma_{\max}}(-z) & \text{if } z \leq 0 \\ D_{\mathbb{Z}^+,\sigma_{\max}}(z-1) & \text{if } z \geq 1. \end{cases} \tag{2.40}$$

Then, to recover the desired distribution $D_{\mathbb{Z},\sigma,\mu}$ for the inputs (σ, μ), one might want to apply the rejection sampling technique presented in Section 2.2.5 and accept z with probability

$$\begin{aligned} \frac{D_{\mathbb{Z},\sigma,\mu}(z)}{BG_{\sigma_{\max}}(z)} &\propto \begin{cases} \exp\left(\frac{z^2}{2\sigma_{\max}^2} - \frac{(z-\mu)^2}{2\sigma^2}\right) & \text{if } z \leq 0 \\ \exp\left(\frac{(z-1)^2}{2\sigma_{\max}^2} - \frac{(z-\mu)^2}{2\sigma^2}\right) & \text{if } z \geq 1 \end{cases} \\ &\propto \exp\left(\frac{z_0^2}{2\sigma_{\max}^2} - \frac{(z-\mu)^2}{2\sigma^2}\right). \end{aligned}$$

The element inside the exp is computed in step 5 of Algorithm 13. Next, we also introduce an algorithm denoted RS.

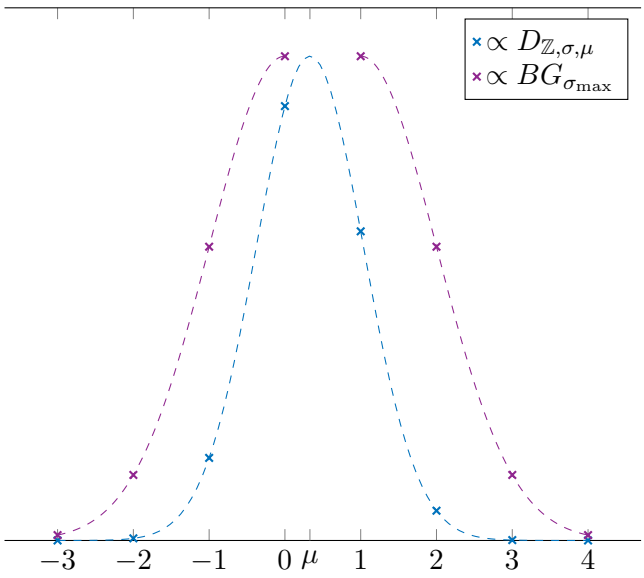


Figure 2.2: *Graphical intuition of the distributions in SamplerZ. For a clear explanation, we have removed the normalization factors, they do not impact the rejection sampling.*

Algorithm 14 — Rejection Sampling for SamplerZ (RS)

Data: $\sigma_{\min} \leq \sigma \leq \sigma_{\max}, x < 0$
Result: a bit b that follows $\mathcal{B}_{\frac{\sigma_{\min}}{\sigma}\cdot\exp(x)}$

1 $p := \frac{\sigma_{\min}}{\sigma} \cdot \exp(x)$
2 $i := 1$ `/* Lazy Bernoulli sampling */`
3 **repeat**
4 $\quad i := i \cdot 2^8$
5 $\quad u \xleftarrow{\$} \{0, 2^8 - 1\}$
6 $\quad v := \lfloor p \cdot i \rfloor$ `& 0xff`
7 **until** $u \neq v$
8 **return** $(u < v)$

The latter performs the rejection sampling (Algorithm 14): using $\exp(\cdot)$, it returns a Bernoulli sample with the according probability. For isochrony matters, detailed in [How+20, Section 6] and highlighted in Section 2.5.5 (in step 1), the acceptance probability is rescaled by a factor $\frac{\sigma_{\min}}{\sigma}$. As z follows the $BG_{\sigma_{\max}}$ distribution, after the rejection sampling, the final distribution of $\mathsf{SamplerZ}(\sigma, \mu)$ is then proportional to $\frac{\sigma_{\min}}{\sigma} \cdot D_{\mathbb{Z},\sigma,\mu}$, which is, after normalization exactly equal to $D_{\mathbb{Z},\sigma,\mu}$.

From steps 2 to 8 in Algorithm 14, we present a lazy comparison. In fact, instead of generating a uniform random value in $[0, 1]$ with a high bit-precision, one can generate the randomness 8 bits by 8 bits and make comparisons on the fly. This technique is not constant-time, but it remains isochronous as the number of repetitions does not leak any information about x.

Thus, with this construction, one can derive the following proposition using Lemma 5.

Proposition 4 (Correctness)**.** *Let us assume that all the uniform distributions are perfect, then the construction of SamplerZ (in Algorithms 13 and 14) is such that* $\mathsf{SamplerZ}(\sigma, \mu) = D_{\mathbb{Z},\sigma,\mu}$.

In practical implementations, one cannot achieve perfect distributions.

Table 2.3: *Number of calls to SamplerZ, BaseSampler and ApproxExp*

	Notation	Value for Falcon
Calls to sign (as per NIST)	$\mathtt{Q}_\mathrm{s}$	$\leq 2^{64}$
Calls to SamplerZ	$\mathtt{Q}_\mathrm{samplZ}$	$\mathtt{Q}_\mathrm{s} \cdot 2 \cdot n \leq 2^{75}$
Calls to $D_{\mathbb{Z}^+,\sigma_\mathrm{max}}$	$\mathtt{Q}_\mathrm{bs}$	$\mathcal{N}_\mathrm{iter} \cdot \mathtt{Q}_\mathrm{samplZ} \leq 2^{76}$
Calls to exp	$\mathtt{Q}_\mathrm{exp}$	$\mathtt{Q}_\mathrm{bs} \leq 2^{76}$

2.5.2 Security statement

Table 2.3 gives the notations for the number of calls to SamplerZ, $D_{\mathbb{Z}^+,\sigma_\mathrm{max}}$, exp and the considered values when the sampler is instantiated for Falcon. Due to the rejection sampling in step 9 in Algorithm 13, there will be a (potentially infinite) number of iterations of the **while** loop. We will show later in Lemma 10, that the number of iterations follows a geometric law of parameter $\approx \frac{\sigma_\mathrm{min}\cdot\sqrt{2\pi}}{2\cdot\rho_{\sigma_\mathrm{max}}(\mathbb{Z}^+)}$. For the example of Falcon, $\frac{\sigma_\mathrm{min}\cdot\sqrt{2\pi}}{2\cdot\rho_{\sigma_\mathrm{max}}(\mathbb{Z}^+)} \leq 2$.

The following Theorem estimates the security of SamplerZ.

Theorem 2 (Security of SamplerZ)**.** *Consider a cryptosystem, denoted C_ideal that makes at most $\mathtt{Q}_\mathrm{s}$ **independent** queries to SamplerZ. Assume that C_ideal is λ-bit secure against a search problem. Let us consider a copy of the latter cryptosystem, denoted C_real, where the only differences are that*

- *each query to $D_{\mathbb{Z}^+,\sigma_\mathrm{max}}$ is replaced by a query to a CDT sampler (defined in Section 2.2.4)[8], denoted BaseSampler*
- *each query to* exp() *is replaced by a query to a polynomial denoted ApproxExp.*

If the two following conditions are respected;

$$\forall x < 0,\ \max\left(\left|\frac{\mathsf{ApproxExp}(x) - \exp(x)}{\exp(x)}\right|, \left|\frac{\mathsf{ApproxExp}(x) - \exp(x)}{1-\exp(x)}\right|\right) \leq \frac{1}{\sqrt{2\cdot(2\cdot\lambda-1)\cdot\mathtt{Q}_\mathrm{exp}}} \quad \text{(Cond. (1))}$$

$$R_{2\cdot\lambda-1}\left(\mathsf{BaseSampler}, D_{\mathbb{Z}^+,\sigma_\mathrm{max}}\right) \leq 1 + \frac{1}{4\cdot\mathtt{Q}_\mathrm{bs}} \quad \text{(Cond. (2))}$$

then, C_real is $(\lambda-2)$-bit secure against the search problem.

Proof: Let us introduce an intermediate case where only the base sampler is modified (the exponential stays a perfect distribution). The 3 cases are defined as follows.

1. (Ideal) an unmodified version of SamplerZ is called
2. (Inter) Only the exponential is considered as perfect.
3. (Real) The modified SamplerZ is called.

We first apply Lemma 2 for assessing the bit security of the Inter case. Thanks to Condition (2), we get that $\lambda_\mathrm{Inter} \geq \lambda - 1$. We then apply Lemma 6 for assessing the bit security of the Real case. Thanks to Condition (1), we get $\lambda_\mathrm{Real} \geq \lambda_\mathrm{Inter} - 1 \geq \lambda - 2$.

□

[8] One slight difference is that the sign is not drawn, thus we replace t_0 with $2t_0$

To get concrete numerical values, we assume that 256 bits are claimed. For an implementation of Falcon, the numerical values are

$$\frac{1}{\sqrt{2\cdot(2\cdot\lambda-1)\cdot \mathtt{Q}_{\mathrm{exp}}}} \approx 2^{-43} \quad \text{and} \quad \frac{1}{4\cdot \mathtt{Q}_{\mathrm{bs}}} \approx 2^{-78}.$$

2.5.3 A polynomial approximation again

To achieve condition (1) with ApproxExp, we use again our GALACTICS tool.

```
sage: load('polynomial_approximation.sage')
....: galactics = GALACTICS_approximation(0, ln(2), 128, 46, 45, "exp(x)")
....: galactics.approximate()
```

```
[...]
 ----------------------
 | Final security : 47  |
 | Max size coeffs : 45 |
 ----------------------
.................................

Exact polynomial:
P = 1 * 2 ^ 0
  + 35184372088831 * 2 ^ -45 * x ^ 1
  + 4398046511129 * 2 ^ -43 * x ^ 2
  + 11728124023891 * 2 ^ -46 * x ^ 3
  + 23456248667481 * 2 ^ -49 * x ^ 4
  + 18764980474255 * 2 ^ -51 * x ^ 5
  + 3127580083557 * 2 ^ -51 * x ^ 6
  + 28579833423385 * 2 ^ -57 * x ^ 7
  + 14399595757739 * 2 ^ -59 * x ^ 8
  + 23649690795601 * 2 ^ -63 * x ^ 9
  + 28643805083125 * 2 ^ -66 * x ^ 10
```

Thanks to an Euclidean division optimization similar to Eq. (2.36), this polynomial is enough to validate condition (1). For validating Eq. (2.24), we use the same technique than in Section 2.4.3 and compute an experimental maximum by discretizing $[0, ln(2)]$. We get

$$\forall x \in [0, \ln(2)], \left| \frac{\exp\left(\frac{x}{2\sigma^2}\right) - P^{I_3}_{\exp}(x)}{1 - \exp\left(\frac{x}{2\sigma^2}\right)} \right| \leq 2^{-45}.$$

Flexibility on the implementation of the polynomial. Depending on the platform and the requirement for the signature, one can adapt the polynomial to fit their constraints. For example, if one wants to minimize the number of multiplications, implementing the polynomial with Horner's form is the best option.

The polynomial is written in the following form:

$$P_1(x) := a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + x(a_5 + x(a_6 + x(a_7 + x(a_8 + x(a_9 + xa_{10}))))))))).$$

Evaluating P_1 is then done serially as follows:

$$\begin{aligned} \mathtt{y} &:= \mathtt{a}_{10} \\ \mathtt{y} &:= \mathtt{a}_9 + \mathtt{y} \times \mathtt{x} \\ &\vdots \\ \mathtt{y} &:= \mathtt{a}_1 + \mathtt{y} \times \mathtt{x} \\ \mathtt{y} &:= \mathtt{a}_0 + \mathtt{y} \times \mathtt{x} \end{aligned}$$

Table 2.4: *Comparison of the polynomial methods for computing* exp

Polynomial	Mults	Adds	Depth	Size of the coefficients
P_1 (Horner)	**10**	10	10	45
P_2 (Estrin)	13	9	**4**	45

Some architectures enjoy some level of parallelism, in which case it is desirable to minimise the depth of the circuit computing the polynomial[9]. Writing P_1 in Estrin's form [Est60] is helpful in this regard. This writing is denoted P_2.

$$\begin{aligned}
\mathtt{x}_2 &\coloneqq \mathtt{x} \times \mathtt{x} \\
\mathtt{x}_4 &\coloneqq \mathtt{x}_2 \times \mathtt{x}_2 \\
\mathtt{P}_3(\mathtt{x}) &\coloneqq (\mathtt{x}_4 \times \mathtt{x}_4) \times ((\mathtt{a}_8 + \mathtt{a}_9 \times \mathtt{x}) + \mathtt{x}_2 \times \mathtt{a}_{10}) \\
&+ (((\mathtt{a}_0 + \mathtt{a}_1 \times \mathtt{x}) + \mathtt{x}_2 \times (\mathtt{a}_2 + \mathtt{a}_3 \times \mathtt{x})) + \mathtt{x}_4 \times ((\mathtt{a}_4 + \mathtt{a}_5 \times \mathtt{x}) + \mathtt{x}_2 \times (\mathtt{a}_6 + \mathtt{a}_7 \times \mathtt{x})))
\end{aligned}$$

In Table 2.4, we present the different methods with their associated number of multiplications, additions, depth of the circuit and size of the coefficients.

2.5.4 An experimentally derived base sampler

To achieve condition (2) with BaseSampler, we rely on a cumulative distribution table (CDT) as described in Section 2.2.4. Instead of providing 128-bit absolute precision to the table entries (as presented in Section 2.4.5), we propose an experimental alternative. We can derive a table that experimentally verifies Condition (1).

Let $a \coloneqq 2 \cdot \lambda - 1$. To derive the parameters w and θ we use a simple script that, given $\sigma_{\max}$ and θ as inputs:

1. Computes the smallest tailcut w such that the Rényi divergence R_a between the ideal distribution $D_{\mathbb{Z}^+,\sigma_{\max}}$ and its restriction to $\{0,\dots,w\}$ (noted $D_{[w],\sigma_{\max}}$) verifies
$$R_a(D_{[w],\sigma_{\max}}, D_{\mathbb{Z}^+,\sigma_{\max}}) \leq 1 + 1/(4\mathtt{Q}_{\mathrm{bs}});$$
2. Rounds the probability density table (PDT) of $D_{[w],\sigma_{\max}}$ with θ bits of absolute precision. This rounding is done "cleverly" by truncating all the PDT values except the largest:
   - for $z \geq 1$, the value $D_{[w],\sigma_{\max}}(z)$ is truncated: $PDT(z) = 2^{-\theta}\left\lfloor 2^{\theta} D_{[w],\sigma_{\max}}(z)\right\rfloor$.
   - in order to have a probability distribution, $PDT(0) = 1 - \sum_{z\geq 1} PDT(z)$.
3. Derives the CDT from the PDT and computes the final $R_a(\mathsf{SampleCDT}_{w=19,\theta=72}, D_{\mathbb{Z}^+,\sigma_{\max}})$.

Taking $\sigma_{\max} = 1.8205$ and $\theta = 72$ as inputs, we found $w = 19$.

- $\mathtt{PDT}(0) = 2^{-72} \times 16976802417466403000 30$
- $\mathtt{PDT}(1) = 2^{-72} \times 14599434566429129596 16$
- $\mathtt{PDT}(2) = 2^{-72} \times 92848835501801105651 5$
- $\mathtt{PDT}(3) = 2^{-72} \times 43669394481705441461 9$
- $\mathtt{PDT}(4) = 2^{-72} \times 15189314079036920101 3$
- $\mathtt{PDT}(5) = 2^{-72} \times 39071441848292237840$
- $\mathtt{PDT}(6) = 2^{-72} \times 7432604049020375675$
- $\mathtt{PDT}(7) = 2^{-72} \times 1045641569992574730$
- $\mathtt{PDT}(8) = 2^{-72} \times 108788995549429682$
- $\mathtt{PDT}(9) = 2^{-72} \times 8370422445201343$
- $\mathtt{PDT}(10) = 2^{-72} \times 476288472308334$
- $\mathtt{PDT}(11) = 2^{-72} \times 20042553305308$
- $\mathtt{PDT}(12) = 2^{-72} \times 623729532807$
- $\mathtt{PDT}(13) = 2^{-72} \times 14354889437$

[9] We are thankful to Thomas Pornin for bringing up this fact.

- $\mathtt{PDT}(14) = 2^{-72} \times 244322621$
- $\mathtt{PDT}(15) = 2^{-72} \times 3075302$
- $\mathtt{PDT}(16) = 2^{-72} \times 28626$
- $\mathtt{PDT}(17) = 2^{-72} \times 197$
- $\mathtt{PDT}(18) = 2^{-72} \times 1$

Our experiment showed that for any $a \geq 511$, $R_a(\mathsf{SampleCDT}_{w=19,\theta=72}, D_{\mathbb{Z}^+,\sigma_{\max}}) \leq 1 + 2^{-80} \leq 1 + \frac{1}{4\mathtt{Q}_{\mathrm{bs}}}$, which validates condition (2) for the Falcon implementation.

2.5.5 Isochrony statement

In this section, we outline the arguments that prevent Algorithm 13 against timing attacks as defined in Security Model 2. One of the differences with the BLISS Gaussian sampler in Section 2.4.5, is that the standard deviation and center *are not* public parameters, and some knowledge on their distribution can lead to attacks (see [Fou+19]). Therefore, we need to formally prove that it is isochronous with respect to the joint distribution of σ, μ and the output z.

To prove that we introduce two intermediary results. We first cite Lemma 10, which shows that the number of iterations in the **while** loop of Algorithm 13 is independent of σ, μ, z. Next, we cite Theorem 3 that outlines an intrinsic property of our sampler: it is perfectly isochronous with respect to z and statistically isochronous (for the Rényi divergence) with respect to σ, μ. We refer to [How+20, Section 6] for the detailed proofs of Lemma 10 and Theorem 3 which Thomas Prest should be credited for.

Lemma 10 (Particular case of Lemma 7 of [How+20]). *For all $\mu \in [0, 1]$ and $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$, the number of iterations of the* **while** *loop in $\mathsf{SamplerZ}(\sigma, \mu)$ (Algorithm 13) follows a geometric law of parameter*

$$P_{\mathbf{true}}(\sigma, \mu) \approx \frac{\sigma_{\min} \cdot \sqrt{2\pi}}{2 \cdot \rho_{\sigma_{\max}}(\mathbb{Z}^+)}.$$

Theorem 3 (Particular case of Theorem 8 of [How+20]). *Let $\mu \in [0, 1]$ and $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$. The running time of $\mathsf{SamplerZ}(\sigma, \mu)$ (Algorithm 13) follows a distribution $T_{\sigma,\mu}$ such that, for any $\epsilon \in (0, 1)$,*

$$R_a(T_{\sigma,\mu}, T) \lesssim 1 + C \cdot a \cdot \epsilon^2$$

where $C > 0$ is a constant and T is a distribution independent of its inputs σ, μ and its output z.

Finally, we introduce a theorem that states the isochrony of a signature using our sampler.

Theorem 4. *Consider an adversary $\mathcal{A}$ making $\mathtt{Q}_{\mathrm{s}}$ signature queries against a signature algorithm that uses $\mathsf{SamplerZ}$. Assume that $\mathcal{A}$ breaks the security by solving a search problem with success probability $2^{-\lambda}$ for some $\lambda \geq 1$. Learning the running time of each call to $\mathsf{SamplerZ}$ does not increase the success probability of $\mathcal{A}$ by more than a constant factor.*

Proof: We denote by D the publicly available signature distribution. We consider two cases:

- Strong: we give powerful timing attack capabilities to the adversary by allowing her to learn the exact runtime of each call to $\mathsf{SamplerZ}(\sigma, \mu)$ without knowing (σ, μ). She then learns the joint distribution $(D, (T_{\sigma_0,\mu_0}, \ldots T_{\sigma_{\mathtt{Q}-1},\mu_{\mathtt{Q}-1}}))$ where $\mathtt{Q}$ is the total number of calls to $\mathsf{SamplerZ}$ (for Falcon $\mathtt{Q} = 2n\mathtt{Q}_{\mathrm{s}}$), $T_{\sigma,\mu}$ denotes the runtime of $\mathsf{SamplerZ}(\sigma, \mu)$ (as in Theorem 3) and (σ_i, μ_i) are the unknown possibly dependent input parameters of $\mathsf{SamplerZ}$.
- Standard: we assume that the attacker has no access to the runtime of each sampling and thus we can write without loss of generality that she has access to $(D, T^{\mathtt{Q}})$ where T is a distribution defined in Theorem 3.

Table 2.5: *Number of samples per second at 2.5 GHz for our sampler and [ZSS19].*

Algorithm	Number of samples
This work[10]	1.84×10^6/sec
This work (AVX2)[11]	7.74×10^6/sec
[ZSS19] (AVX2)[12]	5.43×10^6/sec

Let $P_{\text{STRONG}}, P_{\text{STANDARD}}$ denote the success probability of $\mathcal{A}$ in the STRONG and STANDARD cases, respectively. We set $a := \lambda$ and apply probability preservation.

$$P_{\text{STRONG}} \leq \left(P_{\text{STANDARD}} \cdot R_\lambda((D, (T_{\sigma_0,\mu_0}, \dots T_{\sigma_{\mathtt{Q}-1},\mu_{\mathtt{Q}-1}})), (D, T^{\mathtt{Q}})\right)^{(\lambda-1)/\lambda}. \tag{2.41}$$

Note that D and $(T_{\sigma_0,\mu_0}, \dots T_{\sigma_{\mathtt{Q}-1},\mu_{\mathtt{Q}-1}})$ are independent, thus

$$P_{\text{STRONG}} \leq \left(P_{\text{STANDARD}} \cdot R_\lambda((T_{\sigma_0,\mu_0}, \dots T_{\sigma_{\mathtt{Q}},\mu_{\mathtt{Q}}}), T^{\mathtt{Q}})\right)^{(\lambda-1)/\lambda}. \tag{2.42}$$

Let us replace ϵ by $\frac{1}{\sqrt{\lambda \mathtt{Q}}}$ in Theorem 3 and get

$$\forall \sigma \in [\sigma_{\min}, \sigma_{\max}], \forall \mu \in [0,1] \quad R_\lambda(T_{\sigma,\mu}, T) \lesssim 1 + \frac{C}{\mathtt{Q}}.$$

Therefore, we can apply Proposition 2 and get

$$P_{\text{STRONG}} \lesssim \left(P_{\text{STANDARD}} \cdot \left(1 + \frac{C}{\mathtt{Q}}\right)^{\mathtt{Q}}\right)^{(\lambda-1)/\lambda} \tag{2.43}$$

$$\lesssim 2^{-\lambda} \cdot 2 \cdot e^C. \tag{2.44}$$

□

2.5.6 Applications and limitations

After Thomas Ricosset performed preliminary benchmarks, Thomas Pornin has implemented our sampler as part of the new isochronous implementation of Falcon [Por19]. This implementation can use floating-point hardware or AVX2 instructions when available but also includes floating-point emulation code that uses only usual integer operations. On ARM Cortex M4 CPUs, which can only support single-precision floating-point instructions, this implementation provides assembly implementations for the core double-precision floating-point operations more than twice faster than the generic emulation. As a result, our sampler can be efficiently implemented on embedded platforms as limited as Cortex M4 CPUs, while some other samplers (e.g. [Kar+19] due to a huge code size) are not compact enough to fit embedded platforms.

The benchmarks of this sampler implementation are performed on a single Intel Core i7-6500U CPU core clocked at 2.5 GHz. In Table 2.5, we present the running times of our isochronous sampler in standard double-precision floating-point and in an AVX2 implementation. To compare with [ZSS19], we scale the numbers to be based on 2.5GHz. Note that for our sampler, the number of samples per second is on average for $1.2915 < \sigma \leq 1.8502$ while for [ZSS19] $\sigma = 2$ is fixed.

In Table 2.6, the running times of the Falcon isochronous implementation [Por19] that contains our sampler are presented. One can compare it with a second non-isochronous implementation nearly identical except for the base sampler which is a faster lazy CDT sampler, and the rejection sampling which is not scaled by a constant. Compared to the non-isochronous implementation, the isochronous one is about 22% slower but remains very competitive speed-wise.

[10] [Por19] standard double-precision floating-point (IEEE 754) with SHAKE256.
[11] [Por19] AVX2 implementation wth eight ChaCha20 instances in parallel (AVX2).
[12] [ZSS19] constant-time implementation with hardware AES256 (AES-NI).

Table 2.6: *Falcon signature generation time at 2.5 GHz.*

Degree	**Non-isochronous** (using AVX2)	**Isochronous** (using AVX2)
512	210.88 μs (153.64 μs)	257.33 μs (180.04 μs)
1024	418.76 μs (311.33 μs)	515.28 μs (361.39 μs)

Cache-timing protection. Following this implementation of the proposed sampler also ensures cache-timing protection [Fac+18], as the design *should*[13] bypass conditional branches by using a consistent access pattern (using linear searching of the table) and have isochronous runtime. This has shown to be sufficient in implementations of Gaussian samplers in Frodo [Bos+16; Nae+19].

Advantages and limitations. Our sampler has an acceptance rate $\approx \frac{\sigma_{\min}}{\sigma_{\max}+0.4}$ making it especially suitable when $\sigma_{\min}$ and $\sigma_{\max}$ are close. In particular, our sampler is, so far, the fastest isochronous sampler for the parameters in Falcon. However, the larger the gap between $\sigma_{\min}$ and $\sigma_{\max}$, the lower the acceptance rate. In addition, our sampler uses a cumulative distribution table (CDT) which is accessed in an isochronous way. This table grows while $\sigma_{\max}$ grows, while making both running time and memory usage larger. When $\sigma_{\max}$ is large or far from $\sigma_{\min}$, there exist faster isochronous samplers based on convolution [MW17] and rejection sampling [ZSS19][14] techniques.

Perspective 3 (Application of this sampler to other schemes)**.** *A natural question is how our algorithms presented in Section 2.5.1 could be adapted for other schemes than Falcon, for example [MP12; DLP14; GM18; Ber+18; CGM19]. An obvious bottleneck seems to be the size of the CDT used in* SampleCDT*, which is linear in the standard deviation. For larger standard deviations, where linear searching becomes impractical, convolutions can be used to reduce σ, and thus the runtime of the search algorithm [PDG14; Kha+18]. It would also be interesting to see if the tree-based method of [Kar+19] has better scalability than our CDT-based method, in which case we would recommend it for more significant standard deviations. On the other hand, once the base sampler is implemented, we do not see any obvious obstacle for implementing our whole framework. For example, [CGM19] or using Peikert's sampler [Pei10] (in Falcon) entail a small constant number of standard deviations; therefore the rejection step would be very efficient once a base sampler for each standard deviation is implemented.*

[13] Compilers may alter the design, thus one should always verify the design post-compilation.
[14] The constant-time sampler in [ZSS19] may still reveal σ.

Chapter

3 Masking lattice-based signatures

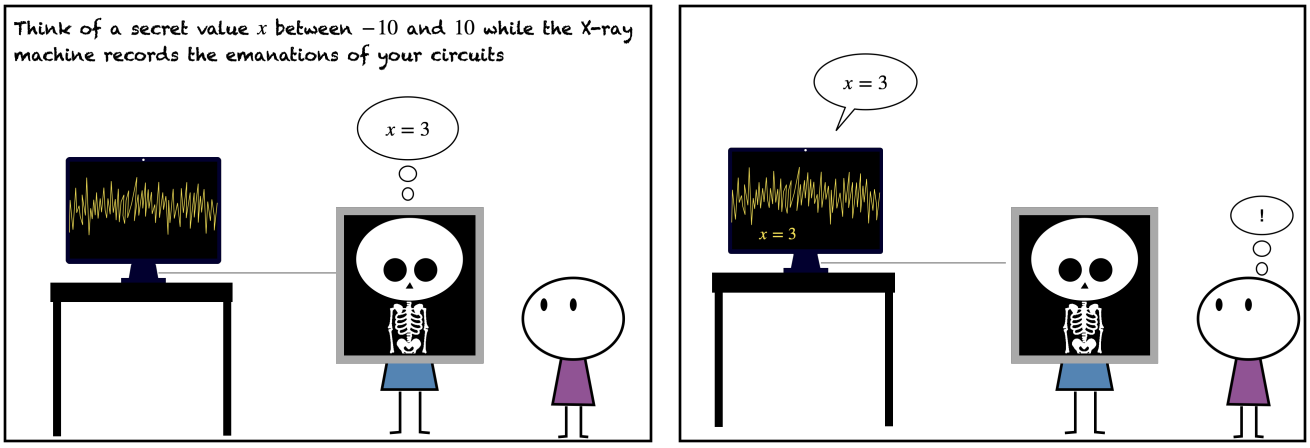


Figure 3.1: *A side-channel attack consists in analyzing physical parameters related to the execution of an algorithm. Here, an X-ray machine records the emanations of the circuits of the blue algorithm. With statistics techniques, one can derive the secret value from the emanations .*

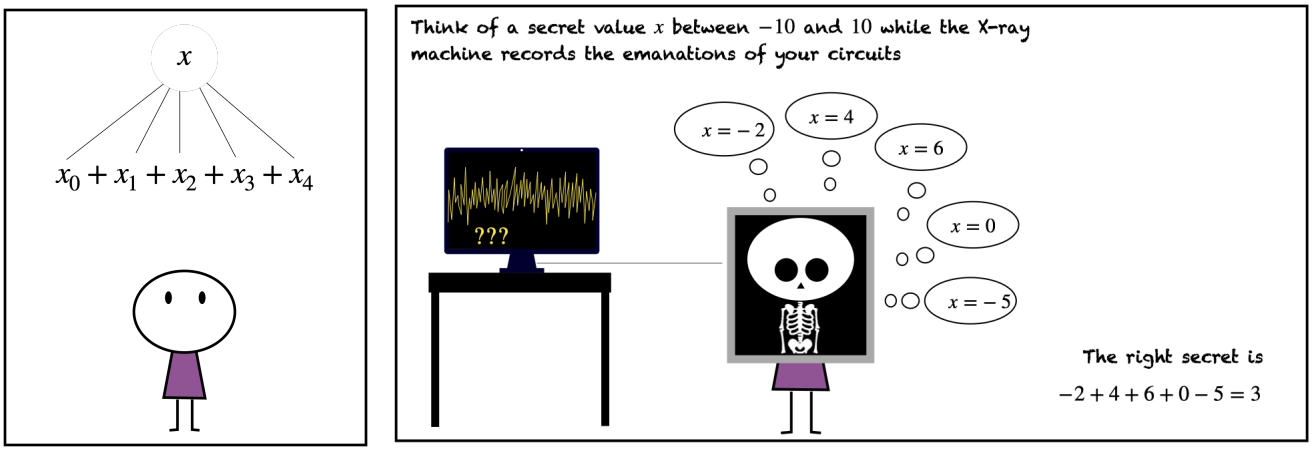


Figure 3.2: *Masking is a countermeasure to prevent side-channel attacks. The purple algorithm adds some noise in its circuits by generating many secrets at the same time while the right secret is the sum of all the values.*

Chapter content

This chapter describes *how to protect lattice-based Fiat–Shamir with aborts signature schemes against side-channel attacks with masking*. Having studied the joys of masking within my first months in Thales alongside Sonia Belaïd, industrial advisor at that time, this topic has been an important part of this PhD. Her PhD was focused on the theoretical side of the masking countermeasure mostly applied to symmetric cryptography, and she wanted to study its behaviour on lattice signatures. It eventually ended up into the creation of the latmasking working group with Sonia Belaïd, Gilles Barthes, Thomas Espitau, Pierre-Alain Fouque, Mehdi Tibouchi and Benjamin Grégoire. In 2017, we masked our first lattice-based signature scheme [GLP12]. Even in the comparatively simple case of GLP, the masking was surprisingly challenging. The probabilistic nature of the signature generation, as well as its reliance on rejection sampling, presented difficulties that had not occurred in earlier schemes. We first wanted to avoid the heavy Boolean/arithmetic conversions *à la* [CGV14], but they turned out to be necessary. More importantly, we noted that, in this type of signature schemes, some of the intermediate values might be revealed to the attacker without threatening the security. The only drawback was that we needed to assume non-standard assumptions. We also had to make a proof of concept to show how the countermeasure scaled, and we benefited Mehdi Tibouchi's fast implementation skills. Eventually, our work was accepted for publication [Bar+18], and we wanted to study other signature schemes that use more contrived distributions.

BLISS [Duc+13] seemed an interesting candidate for this purpose. While dissecting the rejection sampling step of the latter in order to mask it, a new (slight) timing breach has been highlighted. Later, Thomas Espitau and Mehdi Tibouchi set up a new timing attack, and we turned our back to masking at the benefit of protecting BLISS against timing attacks (see Chapter 2). We figured that some of the timing protection techniques could also help for the masking. Eventually, the masking wasn't really the main contribution of the paper which later arose from this work [Bar+19a], but it is, nevertheless, highlighted in this chapter.

Our works led to a different research track started with another PhD student, François Gérard. We wondered about the properties that make a lattice-based signature "masking-friendly". We ended with our ambitious "Maskarade" project where we wanted to design the perfect masking-friendly lattice-based signature. Finally, our scheme was close to the NIST candidate qTESLA [Bin+19](except for its key generation) and more reasonably, we preferred to focus on the latter. It was a very interesting study as it went from the masking proofs and parameter choices to the randomness optimization. François Gérard finally implemented the masked version in a very optimized and portable way, and this ended up in a publication in 2019 [GR19]. In this chapter, as well as in [Bar+18; Bar+19a; GR19], my contribution concerns the theoretical design and proofs of masked algorithms.

3.1 Introduction and motivations

As stated in our introductory chapter (Section 1.1.3), certain cryptographic implementations may be vulnerable to physical attacks in addition to black-box attacks or timing attacks. This family of powerful attacks targets the algorithms' practical implementations. For example, an attacker can observe the device's emanations (e.g. temperature, EM radiation) to recover the secret data. As the demands for practical implementations of postquantum cryptographic schemes get more pressing ahead of the NIST postquantum standardization process, understanding the security of those schemes against physical attacks is of paramount importance. However, the implementation of lattice-based primitives poses new sets of challenges as far as side-channel and other physical attacks are concerned. In particular, the reliance on Gaussian distributions, rejection sampling or the number-theoretic transform for polynomial multiplication has shown to open the door to new types of physical attacks for which it is not always easy to propose efficient countermeasures.

For instance, Groot Bruinderink et al. [Bru+16] demonstrated a cache attack targeting the Gaussian sampling of the randomness used in BLISS signatures [Duc+13], which recovers the entire private key from the side-channel leakage of a few thousand signature generations. Fault attacks have also been demonstrated on all kinds of lattice-based signatures [Esp+16]. In particular, Espitau et al. recover the full BLISS private key using a single fault on the generation of the randomness (and present a similarly efficient attack on GPV-style signatures). Besides, ACM CCS 2017 has featured several papers [Esp+17; PBY17] exposing further side-channel attacks on BLISS, its variant BLISS–B, and their implementation in the strongSwan VPN software. They are based on a range of different side channels (cache attacks, simple and correlation electromagnetic analysis, branch tracing, etc.), and some of them target new parts of the signature generation algorithm, such as the rejection sampling.

This chapter proposes to apply the so-called *masking countermeasure* to lattice-based Fiat–Shamir with aborts signatures to prevent side-channel attacks. It presents the results of [Bar+18], [Bar+19a] and [GR19] in a combined way with a unified framework. We first briefly present the technical context in Sections 3.1.1 and 3.1.2. Then, in Section 3.2, we present the new security notions first introduced in [Bar+18] along with an updated list of gadgets and techniques that are necessary for masking (taking the follow-ups into account). In Section 3.3, we show how they can be applied to GLP. In Section 3.4, we show that this could be extended to BLISS. Finally, in Section 3.5, we present the last masking contribution to the NIST candidate qTESLA.

3.1.1 The masking countermeasure

The *masking* countermeasure, which performs computations on secret-shared data, appears as a natural countermeasure in the side-channel landscape. It is actually the most deployed one. Simultaneously introduced by Goubin and Patarin in [GP99] and Chari *et al.* [Cha+99], it consists of randomizing the sensitive data which depend on the secret values and known variables. Each secret input x is basically split into $d+1$ variables $(x_i)_{0\leq i\leq d}$ referred to as shares: d of them are generated uniformly at random whereas the last one is computed such that their additive combination reveals the secret value x. The integer d is called *masking order*. This technique has been widely applied in prequantum cryptography. Let us introduce two types of additive combinations in the following definition.

Definition 4 (Arithmetic and Boolean masking)**.** *A sensitive value $x \in \mathbb{Z}$ is shared with mod q arithmetic masking if it is split into $d+1$ shares $(x_i)_{0\leq i\leq d}$ with the same bit-size such that*

$$x = x_0 + \cdots + x_d \pmod{q}. \qquad \text{(Arithmetic masking mod } q\text{)}$$

It is shared with Boolean masking if it is split into $d+1$ shares $(x_i)_{0\leq i\leq d}$ with the same bit-size such that

$$x = x_0 \oplus \cdots \oplus x_d. \qquad \text{(Boolean masking)}$$

We sketch here why the masking is a sound technique to prevent side-channel attacks. If the device leaks information on one manipulated variable at a time, a d-order masking generally resists to stronger attacks that can combine information on at most d points of measure. In particular, the authors of [Cha+99] have shown that the number of measurements required to mount a successful differential power consumption attack (presented in Section 1.1.3) increases exponentially with the number of shares. Their model was considering the very classical case where each manipulated bit is leaking the sum of its value and a Gaussian noise. Therefore, the masking order represents a trade-off between security and efficiency. In practice, most masked schemes are in first-order (*i.e.* with two shares) or second-order (*i.e.* with three shares).

While the conceptual idea behind the masking countermeasure is pretty simple, implementing a fully shared algorithm has shown to be a complex task. Although it is straightforward on linear operations on which masking is equivalent to applying the original operation on each share of the sensitive data, the procedure is much more complicated on non-linear functions. Let us first introduce the generic model before explaining how to build proofs.

3.1.2 The ISW model

Theoretical leakage models have been introduced in order to reason correctly on the security of implementations exposed to side-channel attacks. These models followed the observation of

[Cha+99] that the combination of an increasing number of noisy variables exponentially increases the difficulty of an attack. The idea of a model where the attacker receives a fixed number of variables would correspond to a case where the attacker is only able to combine a fixed number of intermediate variables because of the measurement noise.
These models provide proof frameworks for checking that every sensitive intermediate value is always protected. The *probing model* or *ISW model* from its inventors [ISW03] is undoubtedly the most deployed. Throughout all this chapter, we fix $d \geq 0$ corresponding to the masking order.

Definition 5 (ISW model)**.** *A cryptographic implementation is d-probing secure iff the joint distribution of any set of at most d intermediate variables is independent of the secrets.*

Informally, an algorithm is secure in the d-probing model if an attacker has a negligible probability of breaking the security with the knowledge of the public data and with access to d intermediate variables of his choice. This model is convenient to prove the security of an implementation as it manipulates finite sets of exact values. However, at first sight, this model does not seem to reflect the reality of the physical attacks as the latter need noisy leakage models. In 2014, the reduction established in [DDF14] allowed to relate the d-probing security to a side-channel security up to a certain level of noise. We will, therefore, exclusively consider the d-probing security in this thesis.

Signature security property

In this thesis, we consider the masking protection of signature schemes in their general security model. However, the EUF-CMA security definition (as described in Security Model 1 in Section 2.1) does not capture any physical attacker. The goal of the following definition is to design a stronger security property combining EUF-CMA and physical guaranties. It can be seen as a d-probing model counterpart of the "EUF-CMA in the isochronous model" notion introduced in Section 2.2.1 in the context of timing security[1].

Definition 6. *A* KeyUpdate *algorithm is defined as an algorithm that takes a shared private key $(sk_i)_{0\leq i\leq d}$ as input and outputs another shared private key $(sk'_i)_{0\leq i\leq d}$ such that $\sum_i sk_i = \sum_i sk'_i \bmod q$ if the sharing is arithmetic or $\bigoplus_i sk_i = \bigoplus_i sk'_i$ if the sharing is Boolean.*

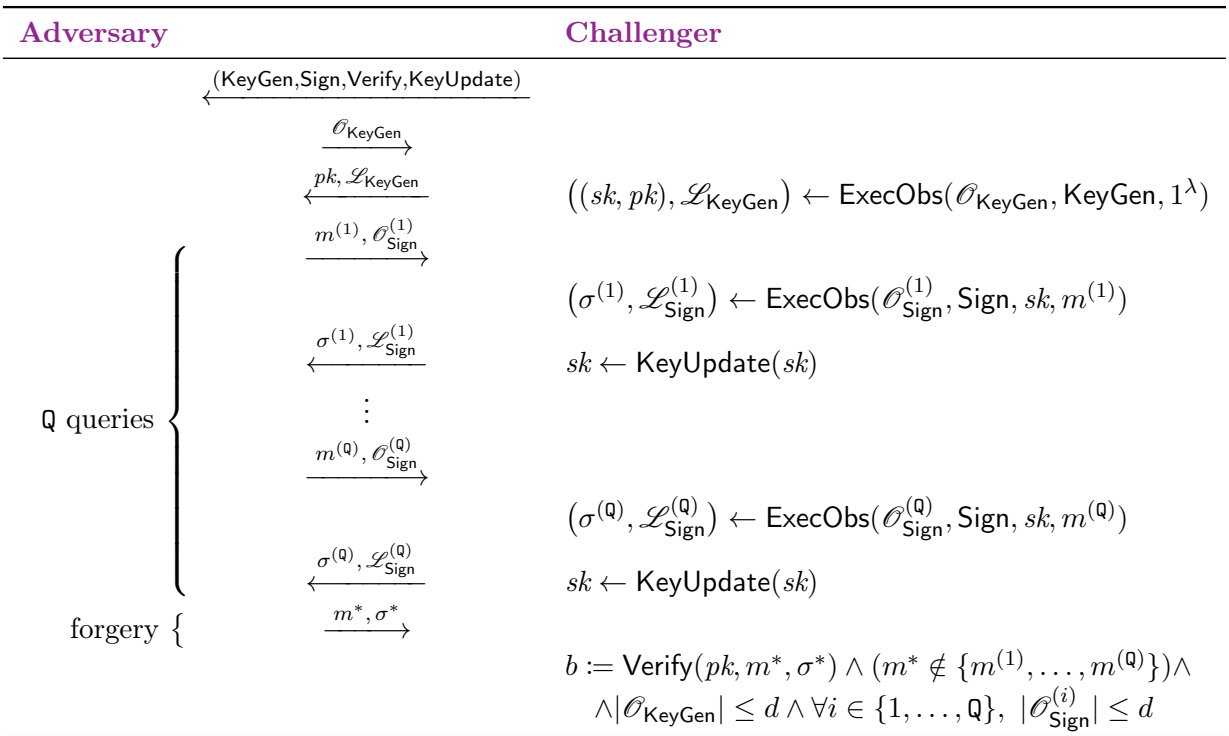
The KeyUpdate algorithm is necessary for the in-between signature queries because an attacker could make d probes for each signature generation which uses $(sk_i)_{0\leq i\leq d}$. Typically, the KeyUpdate will be a naive refreshing (see Algorithm 21).

Security Model 3. *Let $\mathtt{Q}$ be a fixed maximum amount of signature queries. A signature scheme (*KeyGen*, *Sign*, *Verify*) with signing key update algorithm* KeyUpdate *is EUF-CMA-secure in the d-probing model if any probabilistic polynomial time adversary has a negligible probability of winning the Security Game 3.*

If a EUF-CMA signature scheme (KeyGen, Sign, Verify) is such that its algorithms KeyGen$(\cdot)$ and Sign$(\cdot)$ are d-probing secure, according to Definition 5, then one can prove that the signature scheme is EUF-CMA-secure in the d-probing model with a naive refreshing KeyUpdate. This proof will be later detailed for GLP signature scheme in Theorem 8.

Remark 4. *Security Model 3 assumes that the attacker does not get any leakage during the* KeyUpdate*. In this model, the* KeyUpdate *is not part of the signing procedure. Another definition could assume that the attacker also gets leakage during the* KeyUpdate*. In this alternative definition, the total number of observations should be bounded by d. In other words, instead of the condition $\forall i \in \{1,\dots,\mathtt{Q}\},\ |\mathscr{O}^{(i)}_{\mathsf{Sign}}| \leq d$, one should have $\sum_{i\in\{1,\dots,\mathtt{Q}\}} |\mathscr{O}^{(i)}_{\mathsf{Sign}}| \leq d$. Typically, the* KeyUpdate *algorithm will be protected as presented later in Gadget 4.*

[1] This definition was actually introduced in the full version of our GLP paper [Bar+18] in 2018 before the isochronous one [Bar+19a] in 2019.



Security Game 3: *d-probing EUF-CMA game.* ExecObs *is a universal Turing machine that takes as input a series of observations $\mathcal{O}_*$, an algorithm and several possible arguments. It returns the output of the algorithm on those arguments, together with the leakage values $\mathcal{L}_*$ corresponding to the desired observations.*

3.1.3 Probing security and gadgets

Any algorithm is not directly easy to prove d-probing secure according to Definition 5 and Security Model 3. For instance, in non-linear operations, the mixture of shares to compute the result makes it mandatory to introduce random variables and the bigger the program is, the more dependencies to be considered. For example, in [RP10], Rivain and Prouff introduced a masked algorithm SecMult for multiplying two shared integers. Unfortunately, one case was missing in the proof and the latter could not be easily patched. That is why Barthe et al. formally defined in [Bar+16] two security properties, both stronger than Definition 5, namely *non-interference* (Definition 8) and *strong non-interference* (Definition 9). The key idea is to detail and quantify the statement "independent from the secret" of Definition 5 by numbering dependencies in terms of shares of the inputs and outputs. That way, introducing these properties eases the security proofs because (1) one can focus on proving the properties on small parts of the algorithms, and (2) one can securely compose the bricks together and inserting refreshing gadgets (which refresh sharings using fresh randomness) if necessary at carefully chosen locations[2].

[2]Notice that non-interference was already used in practice [RP10; Cor+14] to prove probing security of implementations.

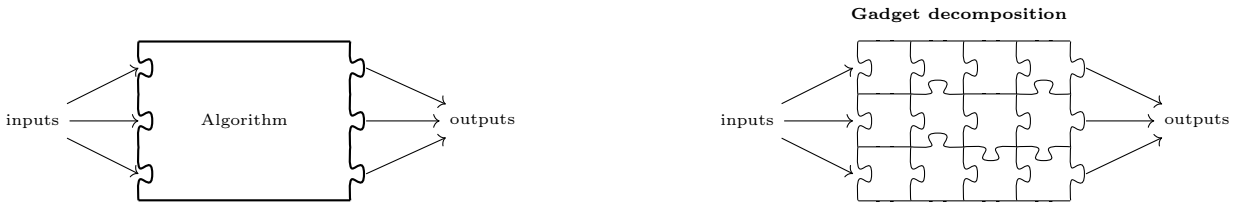


Figure 3.3: *Gadget decomposition*

The main idea of proving the EUF-CMA-security in the d-probing model is first to decompose the algorithm into smaller elementary sub-algorithms, as illustrated in Figure 3.3. Next, study the probing security of each individual gadget and derive its associated properties (see Definition 8 and Definition 9 below). And finally, combine all the properties together by studying the structure of the graph created by the gadgets. The latter is called a *proof of composition.*

Let us first introduce the definition of a gadget.

Definition 7. *A shared value is a $(d+1)$-tuple of values, typically integers.*

A (u, v)-gadget is a probabilistic algorithm that takes as inputs u shared values and returns distributions over v-tuples of shared values. (u, v)-gadgets are used to implement functions that take u inputs and produce v outputs.

Gadgets will be typically written in pseudo-code as they represent sub-parts of larger algorithms. We now turn to the definition of probing security for gadgets.

Definition 8. d-non-interference (d-NI):

A gadget is d-non-interfering if and only if the joint distribution of every set of at most d intermediate variables, corresponding to probes, depends on at most d shares of each input. We also say that every set of at most d intermediate variables can be perfectly simulated *with at most d shares of each input.*

Example 3.1.1. Consider a multiplication algorithm that takes two shared values (a_0, a_1, a_2), $(b_0, b_1, b_2) \in [0, 255]^3$. We want to compute $(c_0, c_1, c_2) \in [0, 255]^3$ such that $c_0 \oplus c_1 \oplus c_2 = (a_0 \oplus a_1 \oplus a_2) \wedge (b_0 \oplus b_1 \oplus b_2)$. Gadget 1 (a particular case of sec& in Gadget 3 that will be described below) is an example of achieving 2-NI security. Let us consider several pairs of intermediate values. The joint distribution of $(r_{0,1}, r_{1,0})$ for instance follows $\{(u, u \oplus a_0b_1 \oplus a_1b_0) : u \xleftarrow{\$} [0, 255]\}$, which only depends on a_0, a_1 and b_0, b_1 (two shares of each input). The joint distribution of $(r_{2,1}, c_0)$ is $\{(u, v) : u, v \xleftarrow{\$} [0, 255]\}$, which does not depend on any of the inputs. Gadget 2 is functionally equivalent but it is not 2-NI secure. The pair $(r_{0,1}, r_{1,0})$ can be simulated with at most two shares of each input whereas the pair $(r_{0,2}, c_2)$ cannot. Indeed, its joint distribution is $\{(u, u \oplus a_0b_2 \oplus a_2b_0 \oplus a_2b_2 \oplus a_1b_2 \oplus a_2b1) : u \xleftarrow{\$} [0, 255]\}$, which depends on 3 shares of each input. It is not even 1-NI because c_1 cannot be simulated with one share of each.

Gadget 1 — Example d-NI ∧

Data: (a_0, a_1, a_2), (b_0, b_1, b_2) $\in [0, 255]^3$
Result: $(c_0, c_1, c_2) \in [0, 255]^3$

1 $r_{0,1}, r_{0,2}, r_{1,2} \xleftarrow{\$} [0, 255]$
2 $r_{1,0} := (r_{0,1} \oplus a_0 b_1) \oplus a_1 b_0$
3 $r_{2,0} := (r_{0,2} \oplus a_0 b_2) \oplus a_2 b_0$
4 $r_{2,1} := (r_{1,2} \oplus a_1 b_2) \oplus a_2 b_1$
5 $c_0 := a_0 b_0 \oplus (r_{0,1} \oplus r_{0,2})$
6 $c_1 := a_1 b_1 \oplus (r_{1,0} \oplus r_{1,2})$
7 $c_2 := a_2 b_2 \oplus (r_{2,0} \oplus r_{2,1})$
8 **return** (c_0, c_1)

Gadget 2 — Example Incorrect ∧

Data: (a_0, a_1, a_2), (b_0, b_1, b_2) $\in [0, 255]^3$
Result: $(c_0, c_1, c_2) \in [0, 255]^3$

1 $r_{0,1}, r_{0,2} \xleftarrow{\$} [0, 255]$
2 $r_{1,0} := (r_{0,1} \oplus a_0 b_1) \oplus a_1 b_0$
3 $r_{2,0} := (r_{0,2} \oplus a_0 b_2) \oplus a_2 b_0$
4 $r_{2,1} := a_1 b_2 \oplus a_2 b_1$
5 $c_0 := a_0 b_0 \oplus (r_{0,1} \oplus r_{0,2})$
6 $c_1 := a_1 b_1 \oplus r_{1,0}$
7 $c_2 := a_2 b_2 \oplus (r_{2,0} \oplus r_{2,1})$
8 **return** (c_0, c_1)

Definition 9. d-strong-non-interference (d-SNI):

A gadget is d-strongly non-interfering if and only if the joint distribution of every set of size $d_0 \leq d$ containing d_1 intermediate variables and $d_2 := d_0 - d_1$ returned values can be perfectly simulated with at most d_1 shares of each input.

We can see that d-SNI implies d-NI, which implies d-probing security if the masking is uniform. The d-SNI property is mostly needed for the refreshing gadgets. The latter are crucial in an implementation as they regenerate the sharing using fresh randomness. We now introduce the affine property for gadgets as introduced in [Bar+16].

Definition 10. *A gadget is affine if and only if it manipulates its input share by share.*

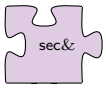
In other words, one observation in an affine gadget can be simulated with only one share of its input. This property will be used for compositions.

The maskComp tool.

For certain composition proofs, we will use the maskComp tool from Barthe et al. [Bar+16]. Mehdi Tibouchi was the one handling this tool. It uses a type-based information flow analysis with cardinality constraints and ensures that the composition of gadgets is d-NI secure at arbitrary orders, by inserting refresh gadgets when required.

Examples of gadgets

Many gadgets are necessary to mask an implementation : multiplication, refreshings, linear operations, unmasking... Let us describe some gadgets existing in the literature. We first introduce a basic gadget that generalizes Gadget 1.



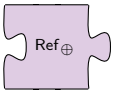
Logical and: This gadget is denoted sec&. It computes the logical and between two values given in a Boolean masked form $(x_i)_{0 \leq i \leq d}$ and $(y_i)_{0 \leq i \leq d}$, the output $(z_i)_{0 \leq i \leq d}$, also in a Boolean masked form is such that $(\bigoplus_i z_i) = (\bigoplus_i x_i) \wedge (\bigoplus_i y_i)$. A masked algorithm has been introduced in [ISW03; RP10] and proved $d - \mathtt{SNI}$ in [Bar+16]. It is detailed in Gadget 3.

Gadget 3 — Bitwise AND of Boolean maskings (sec&)

Data: Boolean maskings $(x_i)_{0\leq i\leq d}$, $(y_i)_{0\leq i\leq d}$ of integers x, y; the bit size w of the masks
Result: A Boolean masking $(z_i)_{0\leq i\leq d}$ of $x \wedge y$

$(z_i)_{0\leq i\leq d} \coloneqq (x_i \wedge y_i)_{0\leq i\leq d}$
for $i = 0$ *to* d **do**
  for $j = i+1$ *to* d **do**
    pick a uniformly random w-bit value r_{ij}
    $r_{ji} \coloneqq (x_i \wedge y_j) \oplus r_{ij}$
    $r_{ji} \coloneqq r_{ji} \oplus (x_j \wedge y_i)$
    $z_i \coloneqq z_i \oplus r_{ij}$
    $z_j \coloneqq z_j \oplus r_{ji}$
  end
end
return $(z_i)_{0\leq i\leq d}$

Let us consider now the refreshing gadget. It has been originally introduced in [ISW03].



Full Refreshing: Refreshes a Boolean sharing $(x_i)_{0\leq i\leq d}$ using fresh randomness. It is denoted $\mathsf{Ref}_{\oplus}$ [Cor14, Algorithm 4] and is made of a succession of $d+1$ linear refresh operations. The output, $(z_i)_{0\leq i\leq d}$, is still such that $(\bigoplus_i z_i) = (\bigoplus_i x_i)$. The details of this gadget are in Gadget 4.

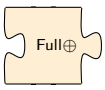
Gadget 4 — Refresh algorithm ($\mathsf{Ref}_{\oplus}$)

Data: A Boolean masking $(x_i)_{0\leq i\leq d}$ of some value x; the bit size w of the returned masks
Result: An independent Boolean masking $(z_i)_{0\leq i\leq d}$ of x

$(z_i)_{0\leq i\leq d} \coloneqq (x_i)_{0\leq i\leq d}$
for $i = 0$ *to* d **do**
  for $j = 1$ *to* d **do**
    pick a uniformly random w-bit value r
    $z_0 \coloneqq z_0 \oplus r$
    $z_j \coloneqq z_j \oplus r$
  end
end
return $(z_i)_{0\leq i\leq d}$

In the literature, mask conversion gadgets exist; these gadgets allow to switch from one type of masking to another type of masking. A first-order masked algorithm of this type has been introduced in [Gou01] and a generic version with order d masked algorithm has been introduced in [CGV14]. We adapted them to the lattice setting where the modulus is not always a power of two and the algorithms are presented in Appendix A.

Finally, we present a gadget that allows to remove the sharing of a data (i.e. to unmask). It is of particular use in signature schemes where the output can be public.



Securely unmasking: This gadget is denoted $\mathsf{Full}\oplus$. It securely refreshes and unmasks an element $(x_i)_{0\leq i\leq d}$ given in Boolean masking form. It outputs z such that $z = (\bigoplus_i x_i)$. This gadget has been introduced and proved $d - \mathtt{NI}$ in [CGV14], it is presented in Gadget 5. We write the security property of $\mathsf{Full}\oplus$ in Lemma 11.

Gadget 5 — Refresh-and-unmask algorithm (Full⊕) from [CGV14]

Data: A Boolean masking $(x_i)_{0\leq i\leq d}$ of some value x; the bit size w of the masks
Result: z, the unmasked value x
1 $(z_i)_{0\leq i\leq d} \leftarrow \mathsf{Ref}_{\oplus}\left((x_i)_{0\leq i\leq d}, w\right)$
2 $z := x_0 \oplus ... \oplus x_d$
3 **return** z

Lemma 11 ([CGV14])**.** Full⊕ *is d-NI secure.*

Proof: Full⊕ is composed of a d-SNI secure refreshing $\mathsf{Ref}_{\oplus}$ (made of $d+1$ linear refreshings inside the loop in Step 2) of the shares and of a Boolean addition of these resulting shares. The attacker is not able to observe an intermediate variable in all the linear refreshings (since she only has $\delta \leq d$ available observations). Thus, we consider that the i^{th} refreshing is left unobserved. As a consequence, all the following observations are independent of x's share except for their sum. That is, Full⊕ is d-NI secure. □

3.2 New techniques for masking lattice-based signatures

In order to protect against physical attacks, one would legitimately like to apply the masking countermeasure to lattice-based cryptography. High order masking can be done quite efficiently and has been applied to the decryption procedure of some lattice-based encryption schemes. Nonetheless, the much more difficult case of lattice-based *signatures* has not been considered until now despite the obvious need for protection. The "masking unfriendly" operations of lattice-based signature schemes are: hard-to-protect implementations of Gaussian distributions and rejection sampling (this technique has been presented in Chapter 2 in Section 2.2.5). For example, the sampling of the Gaussian randomness often involves either very large lookup tables, which are expensive to mask efficiently. The bad news is that most lattice-based signature schemes contain either Gaussian distributions, rejection sampling or both.

3.2.1 A new security property for lattice-based signature schemes

A new security property must be introduced to reason on the security of lattices-based schemes in which some intermediate variables may be revealed to the adversary. Intuitively, a gadget with public outputs O is d-non-interfering with public outputs (d-NIo) iff every set of at most d intermediate variables can be perfectly simulated with at most d shares of each input *and* the public outputs O. The idea here is not to state that public outputs do not leak sensitive information, but rather that the masked implementation does not leak *more* information than the one that is released through public outputs. We capture this intuition by letting the simulator depend on the distribution of the public outputs.

Definition 11. *A gadget with public outputs is a gadget together with a distinguished subset of intermediate variables whose values are broadcasted during execution.*

We now turn to the definition of probing security for gadgets with public outputs.

Definition 12. d-non-interference for gadgets with public outputs(d-NIo):

A gadget with public outputs O is d-NIo if and only if the joint distribution of every set of at most d intermediate variables can be perfectly simulated with the public outputs and at most d shares of each input.

The use of public outputs induces a weaker notion of security.

Lemma 12. *Let G be a d-NI-gadget. Then G is d-NIo secure for every subset O of intermediate variables.*

This lemma states that a gadget that does not leak any information also does not leak *more* information than the one revealed by a subset of its intermediate variables.

Remark 5. *Let G be a d-NIo-gadget as described in Definition 12. When the subset O_G is also part of the returned values, the algorithm is considered d-NI and not d-NIo because the outputs were already available to the attacker in the first place. However, if the gadget G is a part of the composition of another larger gadget G' and if the subset of O_G does not appear in the returned values of G', its unmasked returned values are effectively considered as public outputs.*

Since d-NIo security is weaker than d-NI security, we must justify that it delivers the required security guarantee. This is achieved by combining the proofs of security for a modified version of the signature scheme that returns the public outputs, and the proofs of correctness and security for the masked version. To sum up, for a correct masking proof for lattice-based signature, one needs

- probing security proofs for all the gadgets;
- a proof of composition for the whole algorithm;
- a proof of security for a modified version of the signature scheme that returns the public outputs.

3.2.2 Design of new Gadgets

As shown in the previous section, a complete proof of masking seems tedious to provide as many proofs are necessary. The good news is that many gadgets are applicable to several signature schemes. Therefore, the proof for the gadgets can be done, once and for all, and then be applied to all signature schemes. Thus, only the composition proof remains for each scheme. In the following, we list all the necessary gadgets for masking lattice-based signature schemes. Some gadgets enjoy proofs generated with the maskComp tool [Bar+16]. The other gadgets were easily proved thanks to the composability of the NI and SNI properties.

To avoid overloading this chapter, we do not present all the 14 gadgets that were designed and proved during the thesis. We only present some representative ones in Sections 3.2.3 and 3.2.4. However, a contribution of this thesis is (1) to unify and exhaust all the tools necessary to mask lattice-base Fiat–Shamir with aborts signature and (2) give graphical intuition of the composition proofs. Thus, the other gadgets are still presented in Appendix A. In Table 3.1; we gather all the gadgets introduced during this thesis along with their input and output description, their action, their security property, the reference to their algorithm and to their security proof.

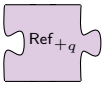
Table 3.1: *Security properties of the gadgets necessary in this thesis.*

Name	Input	Output	Action	Property	Algorithms	Security proof
$\mathsf{Ref}_{\oplus}$	(x_i) Boolean	(z_i) Boolean	$(\bigoplus_i z_i) = (\bigoplus_i x_i)$	$d-\mathtt{SNI}$	Gadget 4	[Cor14, Alg. 4]
Ref_{+_q}	(x_i) Arithmetic	(z_i) Arithmetic	$(\sum_i z_i \mod q) = (\sum_i x_i \mod q)$	$d-\mathtt{SNI}$	see Gadget 4	same as [Cor14, Alg. 4]
$\mathsf{sec}\&$	$(x_i), (y_i)$ Boolean	(z_i) Boolean	$(\bigoplus_i z_i) = (\bigoplus_i x_i) \wedge (\bigoplus_i y_i)$	$d-\mathtt{SNI}$	Gadget 3	[ISW03; RP10; Bar+16]
$\mathsf{Full}\oplus$	(x_i) Boolean	z unmasked	$z = (\bigoplus_i x_i)$	$d-\mathtt{NI}$	Gadget 5	Lemma 11
$\mathsf{Full}+_q$	(x_i) Arithmetic	z unmasked	$z = (\sum_i x_i \mod q)$	$d-\mathtt{NI}$		same as Lemma 11
MAdd	$(x_i), (y_i)$ Arithmetic a unmasked	(z_i) Arithmetic	$\sum_i z_i = a\sum_i x_i + \sum_i y_i \mod q$	$d-\mathtt{NI}$	Gadget 6	Lemma 13
sec_+	$(x_i), (y_i)$ Boolean	(z_i) Boolean	$(\bigoplus_i z_i) = (\bigoplus_i x_i) + (\bigoplus_i y_i)$	$d-\mathtt{NI}$	Gadget 8	Lemma 14
sec_{+_q}	$(x_i), (y_i)$ Boolean	(z_i) Boolean	$(\bigoplus_i z_i) = (\bigoplus_i x_i) + (\bigoplus_i y_i) \mod q$	$d-\mathtt{NI}$	Gadget 7	Lemma 14
$\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$	(x_i) Arithmetic	(z_i) Boolean	$(\bigoplus_i z_i) = (\sum_i x_i \mod q)$	$d-\mathtt{SNI}$	Gadget 13 (simple) Gadget 14 (recur.)	Lemma 23
$\mathsf{B}\!\rightsquigarrow\!\mathsf{A}_q$	(x_i) Boolean	(z_i) Arithmetic	$(\sum_i z_i \mod q) = (\bigoplus_i x_i)$	$d-\mathtt{SNI}$	Gadget 15	Lemma 24
$\mathsf{sec}\lvert.\rvert$	(x_i) Boolean	(z_i) Boolean	$(\bigoplus_i z_i) = \lvert\bigoplus_i x_i\rvert$	$d-\mathtt{NI}$	Gadget 16	Lemma 25
UGen	-	(z_i) Arithmetic	the distribution of $(\sum_i z_i \mod q)$ is uniform	$d-\mathtt{NIo}$	Gadget 9 (int.) Gadget 10 (poly.)	Lemma 15 (int.) Corollary 1 (poly.)
RS	(x_i) Arithmetic $\mathtt{param}$ rs unmasked	unmasked	1 iff $\lvert\sum_i x_i \mod q\rvert \leq \mathtt{param}$, 0 otherwise	$d-\mathtt{NIo}$	Gadget 17 (int.) Gadget 18 (poly.)	Lemma 26 (int.) Corollary 3 (poly.)
Rnd	(x_i) Arithmetic $\mathtt{Tail}$ unmasked	z unmasked	$z = \lfloor\sum_i x_i \mod q\rceil_{\mathtt{Mst}}$ where $\lfloor\cdot\rceil_{\mathtt{Mst}}$ means dropping $\mathtt{Tail}$ least significant bits	$d-\mathtt{NIo}$	Gadget 19 (int.) Gadget 20 (poly.)	Lemma 27 (int.) Corollary 4(poly.)
WRnd	(x_i) Arithmetic $\mathtt{param}_1, \mathtt{param}_2, \mathtt{Tail}$	wr unmasked	1 iff $\lvert\sum_i x_i \mod q\rvert < \mathtt{param}_1$ and $\lvert\lfloor\sum_i x_i \mod q\rceil_{\mathtt{Lst}}\rvert < \mathtt{param}_2$ where $\lfloor\cdot\rceil_{\mathtt{Lst}}$ denotes keeping the $\mathtt{Tail}$ least significant bits	$d-\mathtt{NIo}$	Gadget 21 (int.) Gadget 22 (poly.)	Lemma 28 (int.) Corollary 5 (poly.)
GGen_σ	-	z Arithmetic	the distribution of $(\sum_i z_i \mod q)$ is Gaussian of standard deviation σ	$d-\mathtt{NI}$	Gadget 11 (int.) Gadget 12 (poly.)	Lemma 16 (int.) Corollary 2 (poly.)
MChk	$(\mathbf{x}_i)$ Arithmetic h, S unmasked	ms unmasked	$ms = 1$ iff the sum of the h largest coefficients of $\mathbf{x}$ is larger than S, $ms = 0$ otherwise	$d-\mathtt{NI}$	Gadget 23	Lemma 29

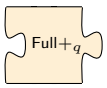
3.2.3 Some new small gadgets

▶ Extend $\mathsf{Ref}_{\oplus}$ and $\mathsf{Full}\oplus$ to arithmetic sharing

In the lattice setting, the sharings are often written in arithmetic mod q masked form. We then extend the definition of $\mathsf{Ref}_{\oplus}$ for this operation.



Full Arithmetic Refreshing: Denoted Ref_{+_q}, this gadget refreshes an arithmetic mod q sharing $(x_i)_{0\le i\le d}$ using fresh randomness. The output, $(z_i)_{0\le i\le d}$, is such that $(\sum_i z_i \mod q) = (\sum_i x_i \mod q)$. The algorithm is the same as Gadget 4 with a $+ \mod q$ in the steps 5 and 6. The security proof holds; it is d-SNI.



Full Addition: This gadget is denoted $\mathsf{Full}+_q$. It sums the shares of a masked arithmetic input $(x_i)_{0\le i\le d}$. The output z is such that $z = (\sum_i x_i \mod q)$. The algorithm is the same as Gadget 5 with a Ref_{+_q} instead of $\mathsf{Ref}_{\oplus}$ and $+ \mod q$ instead of $\oplus$. The security proof holds; it is d-NI.

▶ Multiplication and Addition

Multiply and add: This gadget is denoted MAdd. It performs the addition of masked arithmetic inputs $(x_i)_{0\le i\le d}$ and $(y_i)_{0\le i\le d}$ with a multiplication with an unmasked input a. It outputs $(z_i)_{0\le i\le d}$, arithmetically masked such that $\sum_i z_i = a\sum_i x_i + \sum_i y_i \mod q$. Given in Gadget 6; this algorithm is straightforwardly d-NI as proved in Lemma 13.

Gadget 6 — Multiplication and Addition (MAdd)

Data: a, $(x_i)_{0\le i\le d}$, $(y_i)_{0\le i\le d}$
1 **for** $0 \le i \le d$ **do**
2 $\quad z_i \coloneqq ax_i + y_i$
3 **end**
4 **return** $(z_i)_{0\le i\le d}$

Lemma 13. *The* MAdd *gadget is d-NI secure.*

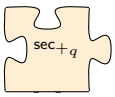
Proof: Let $\delta \le d$ be the number of observations made by the attacker. The only possible observations are the sensitive values x, y and z. We would like to consider all the possible d-uple of intermediate variables. To be exhaustive, the proof consists in filling an empty set $\mathbf{I}$ with at most δ indices in $[0, d]$ such that the distribution of any tuple $(v_1, ..., v_\delta)$ of intermediate variables of the block can be perfectly simulated from the sensitive values.

We build the set $\mathbf{I}$ as follows: For each $v \in (v_1, ..., v_\delta)$, there exists $i \in [0, d]$ such that the value v can be written of the form z_i, x_i, ax_i or y_i and we include such a i in $\mathbf{I}$. The later set contains thus at most δ indices.

Besides, by construction, we can simulate each $v \in (v_1, ..., v_\delta)$ with the corresponding share of the inputs. In the end, any set of $\delta \le d$ intermediate variables can be perfectly simulated with at most δ shares of each sensitive input. This is enough to prove that MAdd is d-NI secure.

□

▶ Modular addition



Modular addition: This gadget is denoted sec_{+_q}. It performs the addition mod q of two Boolean masked inputs $(x_i)_{0\le i\le d}$ and $(y_i)_{0\le i\le d}$. It outputs $(z_i)_{0\le i\le d}$ such that $(\bigoplus_i z_i) = (\bigoplus_i x_i) + (\bigoplus_i y_i) \mod q$. The algorithm is given in Gadget 7, it uses another Gadget 8 as a subroutine.

Gadget 7 — Mod-q addition of Boolean maskings (sec_{+_q})

Data: Boolean maskings $(x_i)_{0\le i\le d}$, $(y_i)_{0\le i\le d}$ of integers x, y; the bit size w of the masks (with $2^w > 2q$)
Result: A Boolean masking $(z_i)_{0\le i\le d}$ of $x + y \bmod q$

if *q is a power of two* **then**
  $(z_i)_{0\le i\le d} \leftarrow \mathsf{sec}_+\big((x_i)_{0\le i\le d}, (y_i)_{0\le i\le d}, w\big)$
  return $(z_i \wedge (q-1))_{0\le i\le d}$
else
  $(q_i)_{0\le i\le d} \coloneqq \big(2^w - q, 0, \ldots, 0\big)$
  $(s_i)_{0\le i\le d} \leftarrow \mathsf{sec}_+\big((x_i)_{0\le i\le d}, (y_i)_{0\le i\le d}, w\big)$
  $(s'_i)_{0\le i\le d} \leftarrow \mathsf{sec}_+\big((s_i)_{0\le i\le d}, (q_i)_{0\le i\le d}, w\big)$
  $(b_i)_{0\le i\le d} \coloneqq \big((s'_i \gg (w-1))\big)_{0\le i\le d}$
  $(b_i)_{0\le i\le d} \leftarrow \mathsf{Ref}_{\oplus}\big((b_i)_{0\le i\le d}, w\big)$
  $(z_i)_{0\le i\le d} \leftarrow \mathsf{sec}\&\big((s_i)_{0\le i\le d}, (b_i)_{0\le i\le d}, w\big)$
  $(b_i)_{0\le i\le d} \leftarrow \mathsf{Ref}_{\oplus}\big((b_i)_{0\le i\le d}, w\big)$
  $(z_i)_{0\le i\le d} \leftarrow (z_i)_{0\le i\le d} \oplus \mathsf{sec}\&\big((s'_i)_{0\le i\le d}, (\neg b_i)_{0\le i\le d}, w\big)$
  return $(z_i)_{0\le i\le d}$
end

Gadget 8 — Integer addition (sec_+)

Data: Boolean maskings $(x_i)_{0\le i\le d}$, $(y_i)_{0\le i\le d}$ of integers x, y; the bit size w of the masks
Result: A Boolean masking $(z_i)_{0\le i\le d}$ of $x + y$

$(p_i)_{0\le i\le d} \coloneqq (x_i \oplus y_i)_{0\le i\le d}$
$(g_i)_{0\le i\le d} \leftarrow \mathsf{sec}\&\big((x_i)_{0\le i\le d}, (y_i)_{0\le i\le d}, w\big)$
for $j = 1$ *to* $W := \lceil \log_2(w-1) \rceil - 1$ **do**
  $\mathrm{pow} := 2^{j-1}$
  $(a_i)_{0\le i\le d} \coloneqq (g_i \ll \mathrm{pow})_{0\le i\le d}$
  $(a_i)_{0\le i\le d} \leftarrow \mathsf{sec}\&\big((a_i)_{0\le i\le d}, (p_i)_{0\le i\le d}, w\big)$
  $(g_i)_{0\le i\le d} \coloneqq (g_i \oplus a_i)_{0\le i\le d}$
  $(a'_i)_{0\le i\le d} \coloneqq (p_i \ll \mathrm{pow})_{0\le i\le d}$
  $(a'_i)_{0\le i\le d} \leftarrow \mathsf{Ref}_{\oplus}\big((a'_i)_{0\le i\le d}, w\big)$
  $(p_i)_{0\le i\le d} \leftarrow \mathsf{sec}\&\big((p_i)_{0\le i\le d}, (a'_i)_{0\le i\le d}, w\big)$
end
$(a_i)_{0\le i\le d} \coloneqq (g_i \ll 2^W)_{0\le i\le d}$
$(a_i)_{0\le i\le d} \leftarrow \mathsf{sec}\&\big((a_i)_{0\le i\le d}, (p_i)_{0\le i\le d}, w\big)$
$(g_i)_{0\le i\le d} \coloneqq (g_i \oplus a_i)_{0\le i\le d}$
$(z_i)_{0\le i\le d} \coloneqq \big(x_i \oplus y_i \oplus (g_i \ll 1)\big)_{0\le i\le d}$
return $(z_i)_{0\le i\le d}$

For the non-power of two particular case, the approach is to first compute $(s_i) = \mathsf{sec}_+\big((x_i),(y_i)\big)$, which is a Boolean sharing of the sum $s = x + y$ without modular reduction, and then $(s'_i)_{0\le i\le d} = \mathsf{sec}_+\big((s_i)_{0\le i\le d},(q_i)_{0\le i\le d}\big)$ for a Boolean masking $(q_i)_{0\le i\le d}$ of the value $-q$ in two's complement form (or equivalently $2^w - q$). The result is a masking of $s' = s - q$ in two's complement form. In particular, we have $s \ge q$ if and only if the most significant bit b of s' is 0. Denote by z the desired modular addition $x + y \bmod q$. We thus have:

$$z = \begin{cases} s & \text{if } b = 1; \\ s' & \text{if } b = 0. \end{cases}$$

As a result, we can obtain the masking of z in a secure way as:

$$(z_i)_{0\le i\le d} = \mathsf{sec}\&\big((s_i)_{0\le i\le d},(b_i)_{0\le i\le d}\big) \oplus \mathsf{sec}\&\big((s'_i)_{0\le i\le d},(\neg b_i)_{0\le i\le d}\big).$$

We consider signed integers, so the $\gg$ in step 4 is an arithmetic shift and actually writes the sign bit in the whole register. This concludes the description of sec_{+_q}.

Lemma 14. *Gadgets* sec_+ *and* sec_{+_q} *are d-NI secure.*

Proof: Gadget sec_+ is built from the Kogge-Stone adder of [Cor+15] with secure AND and secure linear functions such as exponentiations and Boolean additions. As to ensure its security with the combination of these masked atomic functions, the tool maskComp was used to insert the mandatory d-SNI refreshings ($\mathsf{Ref}_\oplus$) properly. As deeply explained in its original paper, maskComp provides a formally proven d-NI secure implementation.

Gadget sec_{+_q} is built from the gadget sec_+ and sec& and linear operations (like $\oplus$). We also use the tool maskComp to generate a verified implementation automatically. Note that the tool automatically adds the two refreshes (lines 9 and 11) and provides a formally proven d-NI secure implementation.

□

Remark 6. *This arbitrary modulus version is quite expensive when q is not a power of two as it uses* sec_+, $\mathsf{Ref}_\oplus$ *and* sec& *twice. Nevertheless, the modulus of the arithmetic masking is fixed by the scheme parameters. Then, for a "masking friendly" design, we will see that choosing a power of two for the modulus when it is possible allows a significant speedup. In Section 3.3, we keep the non-power of two moduli, and in Section 3.5 we make such a trade-off.*

3.2.4 Some new large gadgets

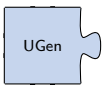
In this subsection, we define gadgets that can be generalized for polynomials in R_q (defined in Eq. (1.3)) where q is a modulus and n is an integer. For a polynomial $p \in R_q$, we will denote by $p^{(i)}$ its i-th coefficient represented in a centered manner $p^{(i)} \in \{-q/2, \dots q/2-1\}$.

Definition 13. *For an integer k such that $0 < k \le (q-1)/2$, we denote by $R_{q,k}$ the elements of R_q (defined in Eq. (1.3)) with coefficients in the range $[-k, k]$.*

▶ Generate a polynomial uniformly in a ring

In many lattice-based signature schemes, polynomials are sampled such that their coefficients are sampled uniformly and independently from an integer interval of the form

$[-k, k]$. In most cases, one would like to obtain those values in masked form, using order-d arithmetic masking modulo q. Note that since all of these coefficients are completely independent, the problem reduces to obtaining an order-d mod-q arithmetic masking of a single random integer in $[-k, k]$Accordingly, we will first create an algorithm called UGen-Coeff, which generates an order-d mod-q arithmetic masking of a single random integer in $[-k, k]$. Next, we will use UGen-Coeff in an algorithm called UGen, which generates a sharing of a value in R_q (defined in Eq. (1.3)). UGen is calling UGen-Coeff n times and is described in Gadget 10. UGen-Coeff is described hereafter and will be given in Gadget 9. The design of this gadget has been improved in this thesis since its paper version [Bar+18].



Uniform Generation: This gadget denoted UGen-Coeff in its coefficient version and UGen in its polynomial version outputs $(z_i)_{0\leq i\leq d}$ in mod q arithmetic masked form such that the distribution of $(\sum_i z_i \mod q)$ is uniform. It is described in Gadget 9 and Gadget 10.

Gadget 9 — Uniform data generation for integers (UGen-Coeff)

Data: k
Result: A uniformly random integer in $[-k, k]$ in mod-q arithmetic masked form $(z_i)_{0\leq i\leq d}$.

1 Let w_0 be the smallest integer such that $2^{w_0} > |K|$ where $K := -2k - 1$
2 Generate uniformly random w_0-bit values $(x_i)_{0\leq i\leq d}$
3 initialize $(k_i)_{0\leq i\leq d} := (K, 0, \cdots, 0)$
4 $(\delta_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+\big((x_i)_{0\leq i\leq d}, (k_i)_{0\leq i\leq d}\big)$
5 $(b_i)_{0\leq i\leq d} := (\delta_i)_{0\leq i\leq d} \gg (w_0 - 1)$
6 $b := \mathsf{Full}\oplus\big((b_i)_{0\leq i\leq d}\big)$
7 **if** $b = 0$ **then**
8 | **Goto** 2
9 **end**
10 $z_i \leftarrow \mathsf{B}\rightsquigarrow\mathsf{A}_q((x_i)_{0\leq i\leq d}, q)$
11 $z_0 := a_0 - k$
12 **return** $(z_i)_{0\leq i\leq d}$

Gadget 10 — Uniform data generation (UGen)

Data: k and d
Result: A uniformly random polynomial $\mathbf{z}$ in $R_{q,k}$ in arithmetic masked form $(\mathbf{z}_i)_{0\leq i\leq d}$.

1 $(\mathbf{z}_i)_{0\leq i\leq d} := \{0\}^d$
2 **for** $j = 0$ *to* $n - 1$ **do**
3 | $(a_i)_{0\leq i\leq d} \leftarrow$ UGen-Coeff(k, d)
4 | $(\mathbf{z}_i)_{0\leq i\leq d} := (\mathbf{z}_i + a_i X^j)_{0\leq i\leq d}$
5 **end**
6 **return** $(\mathbf{z}_i)_{0\leq i\leq d}$

Lemma 15. *Gadget* UGen-Coeff *is d-NIo secure with public output b.*

Proof: Here we need to ensure that the returned shares of z cannot be revealed to the attacker through a d-order side-channel attack. The gadget xGen represents Step

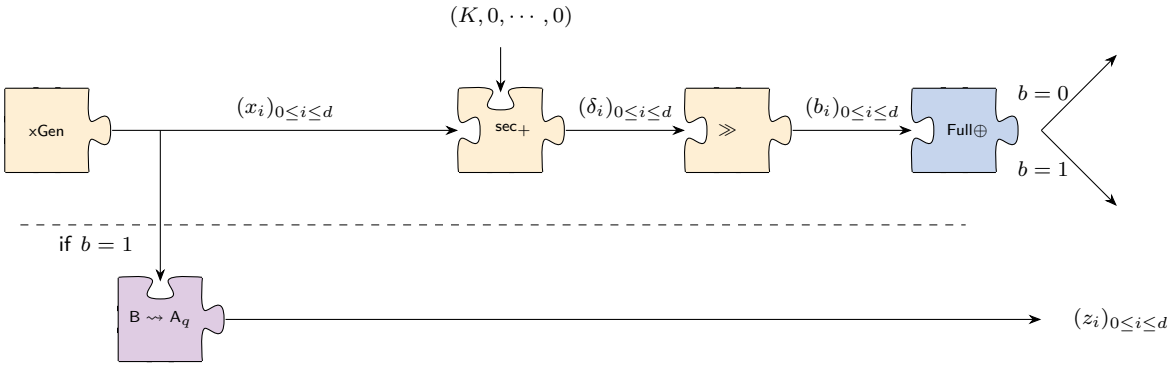


Figure 3.4: *Randomness Generation* UGen-Coeff

2. It is just a random generation of shares. The idea is to prove that any set of $\delta \leq d$ observations on UGen-Coeff can be perfectly simulated with at most δ shares of x.

Gadget UGen-Coeff is built with no cycle. In this case, from the composition results of [Bar+16], it is enough to prove that each sub-gadget is d-NI to achieve global security.

From Lemmas 14 and 23, sec$_+$ and A$_q$$\rightsquigarrow$B are d-NI secure. $\gg$ is trivially d-NI secure as well since it applies a linear function. Besides, gadget Full$\oplus$ is d-NI secure with Lemma 11, and its output is unmasked and thus considered as a public output of UGen-Coeff.

□

Corollary 1. *Gadget* UGen *is d-NIo secure with public output b.*

Proof: From Lemma 15, Gadget UGen is d-NIo secure since it only consists of the linear application of Gadget UGen-Coeff to build the polynomial coefficients. □

▶ Fixed center, fixed standard deviation Gaussian Generation

This gadget is needed for generating Gaussian samples. We mask the technique of cumulative distribution table (CDT) explained in Chapter 2 in Section 2.2.4 with $D = |D_{\mathbb{Z},\sigma}|$. We recall that it consists in precomputing a table of the cumulative distribution function of $|D_{\mathbb{Z},\sigma}|$ with a certain precision θ. To produce a sample, we generate a random value in $(0, 1]$ with the same precision, and return the index of the last entry in the table that is smaller than that value.

Gaussian Generation: This gadget denoted GGen$_\sigma$-Coeff in its coefficient version and GGen$_\sigma$ in its polynomial version outputs $(z_i)_{0\leq i\leq d}$ such that the distribution of $(\sum_i z_i \mod q)$ is Gaussian of standard deviation σ. It is presented in Gadget 11 and Gadget 12.

We present the masked version of table lookup in Gadget 11. The parameters w and θ are respectively the number of elements of the table and the bit precision of its entries. Gadget 11 follows Algorithm 2 in a masked form. The steps 4 to 6 compute $(b_i)_{0\leq i\leq d}$ a masked form of the bit representing $(t_j > u)$. The steps 7 to 10 compute $(z_i)_{0\leq i\leq d}$ a Boolean masked form of $(z + b_i)_{0\leq i\leq d}$ (where $(z_i)_{0\leq i\leq d}$ is initiated to 0). Recall, as noted

in Section 2.2.4, that this algorithm samples only half of the Gaussian, this means that to get a centered distribution, one should sample random signs for each coefficient afterwards.

Gadget 11 — Gaussian Generation for integers (GGen_σ-Coeff)

Data: A table of w probability values t_j with θ bits of precision.
Result: An arithmetic masking of an element $(z_i)_{0\leq i\leq d}$ following a Gaussian of standard deviation σ

$(z_i)_{0\leq i\leq d} \coloneqq (0,\dots,0)$
initialize $(u_i)_{0\leq i\leq d}$ as a θ-bit Boolean masking of a uniform random value $r \in (0,1]$
for $0 \leq j \leq w$ **do**
  initialize $(k_i)_{0\leq i\leq d}$ as a θ-bit Boolean masking of $-t_j$
  $(\delta_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+\big((u_i)_{0\leq i\leq d}, (k_i)_{0\leq i\leq d}\big)$
  $(b_i)_{0\leq i\leq d} \coloneqq (\delta_i)_{0\leq i\leq d} \gg (\theta-1)$
  $(b'_i)_{0\leq i\leq d} \leftarrow \mathsf{sec\&}\,((b_i)_{0\leq i\leq d}, (z_i)_{0\leq i\leq d})$
  initialize $(J_i)_{0\leq i\leq d}$ as a θ-bit Boolean masking of the index j
  $(b_i)_{0\leq i\leq d} \leftarrow \mathsf{sec\&}\,(\neg(b_i)_{0\leq i\leq d}, (J_i)_{0\leq i\leq d})$
  $(z_i)_{0\leq i\leq d} \coloneqq (b'_i)_{0\leq i\leq d} \oplus (b_i)_{0\leq i\leq d}$
end
return $\mathsf{B}\rightsquigarrow\mathsf{A}_q((z_i)_{0\leq i\leq d})$

Remark 7. *Due to the table, this process is quite heavy. For the table's size optimization, a Rényi divergence technique is often used. It consists in using an upper bound on the relative error between the CDT distribution and an ideal Gaussian. This upper bound decreases with the maximum number of queries to the algorithm and it is often lower than a statistical distance estimation. Thus, with Rényi divergence techniques, smaller tables allow the same bits of security. Interestingly, if* GGen_σ *is part of the key generation, the latter cannot be applied. Firstly, the maximum amount of queries to the key generation is not clearly bounded. Secondly, the key generation's security is often based on a decisional problem, and it is currently an open problem to apply the Rényi divergence techniques to decisional problems. Another possible optimization to reduce the size of the table would be to use the recursivity provided in [MW17].*

Lemma 16. *The gadget* GaussGen *in Gadget 12 is d-NI secure.*

Proof: A graphical representation of Gadget 12 is in Fig. 3.5. First, assume that each iteration is d-SNI. Then the algorithm is a linear succession of d-SNI gadgets. The only subtlety is that the element (u_i) forms cycles that are broken by the d-SNI property of each iteration.
Now let us show that each iteration denoted iter is d-SNI. The result comes from the structure of this sub-gadget: the dependencies of the observations on the returned values are stopped by the d-SNI property of the sec& gadget.

□

As with the previous large gadgets, the GGen_σ-Coeff gadget can be generalized for polynomials in R_q by applying GGen_σ-Coeff to each of the coefficients. We present the polynomial version in Gadget 12. From Lemma 16, we can make the following corollary.

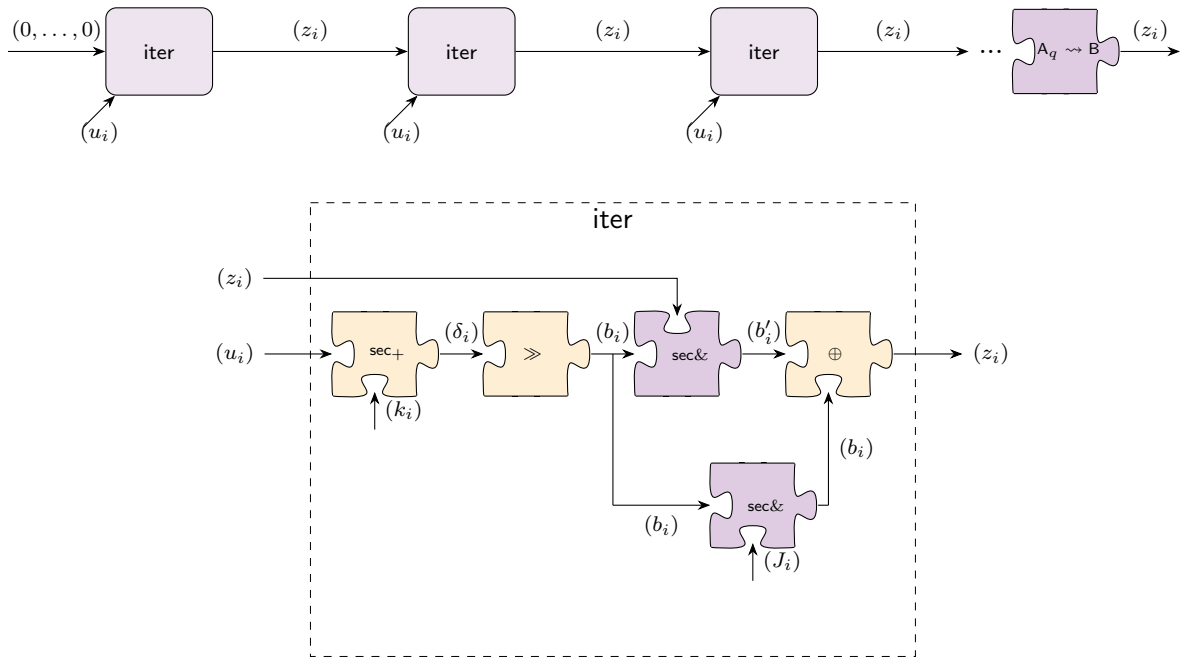


Figure 3.5: *Masked* GGen_σ-Coeff *structure*

Corollary 2. *Gadget* GGen_σ *is d-NI secure.*

Gadget 12 — Gaussian Generation (GGen_σ)

Data: A table of w probability values p_j with θ bits of precision.
Result: An arithmetic masking of a polynomial $(\mathbf{z}_i)_{0\leq i\leq d}$ whose coefficients follow a Gaussian of standard deviation σ

1 **for** $j = 0$ *to* $n-1$ **do**
2 $\quad z^{(j)} \leftarrow \mathsf{GGen}_\sigma\text{-Coeff}()$
3 **end**
4 **return** $\mathbf{z} := \sum_i z^{(i)} \cdot X^i$

3.3 Application to GLP signature scheme

As shown for Gadget 12, masked Gaussian sampling seems very inefficient. Nevertheless, there exist lattice-based signatures that appear to support side-channel countermeasures like masking more naturally, because they entirely avoid Gaussians and other contrived distributions. Both randomness and rejection samplings target uniform distributions in contiguous intervals. Examples of such schemes include the GLP scheme of Güneysu, Lyubashevsky and Pöppelmann [GLP12], which can be seen as the ancestor of BLISS, and later variants like the Dilithium scheme of Ducas et al. [Lyu+19].

3.3.1 The signature scheme

For GLP signature scheme, we will use the following parameters: n is a power of 2, q is a prime number congruent to 1 modulo $2n$ and R_q is defined with Eq. (1.3). The elements of R_q can be represented by polynomials of degree $n-1$ with coefficients in the range $\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2} - 1\}$. We also use Definition 13 notation $R_{q,k}$.

The key generation algorithm for the GLP signature scheme is as follows:

Algorithm 15 — GLP key generation

Result: Signing key sk, verification key pk

1 $\mathbf{s_1}, \mathbf{s_2} \xleftarrow{\$} R_{q,1}$ `/* s1 and s2 have coefficients in {−1, 0, 1} */`
2 $\mathbf{a} \xleftarrow{\$} R_q$
3 $\mathbf{t} \coloneqq \mathbf{as_1} + \mathbf{s_2}$
4 $sk \coloneqq (\mathbf{s_1}, \mathbf{s_2})$
5 $pk \coloneqq (\mathbf{a}, \mathbf{t})$

Given the verification key $pk = (\mathbf{a}, \mathbf{t})$, if an attacker can derive the signing key, this can also be used to solve a $\mathbf{DCK}_{q,n}$ problem defined in [GLP12].

Hard Problem 8 — Decisional Compact Knapsack problem (DCK)

Let q, n be integer parameters.

Given a sample $(\mathbf{a}, \mathbf{b}) \in R_q \times R_q$

Distinguish whether this sample has been drawn from the uniform distribution over $R_q \times R_q$ or the distribution $(\mathbf{a}, \mathbf{as}_1 + \mathbf{s}_2)$ with $\mathbf{a}$ uniformly random in R_q and $\mathbf{s}_1$,$\mathbf{s}_2$ uniformly random in $R_{q,1}$.

Let us now describe the signature scheme introduced in [GLP12]. Additional functions like transform and compress introduced in [GLP12] can be used to shorten the size of the signatures. Note, however, that for masking purposes, we only need to consider the original, non-compressed algorithm of Güneysu et al., which we describe below. Indeed, signature compression does not affect our masking technique at all, because it only involves unmasked parts of the signature generation algorithm (the input of the hash function and the returned signature itself). As a result, although this work only discusses the non-compressed scheme, we can directly apply our technique to the compressed GLP scheme with no change. In fact, this is what Mehdi Tibouchi's proof-of-concept implementation in Section 3.3.5 actually does. The signature scheme needs a particular cryptographic hash function,

$$\mathsf{hash} : \{0,1\}^* \to \mathcal{D}_\alpha^n, \tag{3.1}$$

where $\mathcal{D}_\alpha^n$ is the set of polynomials in R_q that have all zero coefficients except for at most $\alpha = 32$ coefficients that are in $\{-1, +1\}$ (or $\alpha = 16$ when using the updated parameters presented in [Cho17]).

Let κ be a parameter. Algorithm 16 and Algorithm 17, respectively describe the GLP signature and verification. Here is the soundness equation for the verification :

$$\mathbf{az}_1 + \mathbf{z}_2 - \mathbf{tc} = \mathbf{ay}_1 + \mathbf{y}_2.$$

Algorithm 16 — GLP signature

Data: m, $pk = (\mathbf{a}, \mathbf{t})$, $sk = (\mathbf{s}_1, \mathbf{s}_2)$
Result: Signature sig

$\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} R_{q,\kappa}$
$\mathbf{c} \coloneqq \mathsf{hash}(\mathbf{r} \coloneqq \mathbf{a}\mathbf{y_1} + \mathbf{y_2}, m)$
$\mathbf{z}_1 \coloneqq \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$
$\mathbf{z}_2 \coloneqq \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$
if $\mathbf{z}_1$ *or* $\mathbf{z}_2 \notin R_{q,\kappa-\alpha}$ **then**
  restart
end
return $\mathsf{sig} \coloneqq (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

Algorithm 17 — GLP verification

Data: m, $\mathsf{sig} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$, $pk = (\mathbf{a}, \mathbf{t})$
if $\mathbf{z}_1$, $\mathbf{z}_2 \in R_{q,\kappa-\alpha}$ *and* $\mathbf{c} = \mathsf{hash}(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, m)$ **then**
  accept
else
  reject
end

The parameter κ controls the trade-off between the security and the runtime of the scheme. The smaller κ gets, the more secure the scheme becomes and the shorter the signatures get but the time to sign increases. The authors of the implementation of [GLP12] suggest $\kappa = 2^{14}$, $n = 512$, $\alpha = 32$ and $q = 8383489$ for ≈ 100 bits of security and $\kappa = 2^{15}$, $n = 1024$, $\alpha = 32$ and $q = 16760833$ for > 256 bits of security.

3.3.2 Security proof of the r-GLP variant

One of the operations in the signature generation is the computation of a hash function mapping to polynomials in R_q of a very special shape (being in $\mathcal{D}^n_\alpha$ in Eq. (3.1)). Masking the computation of this hash function would be highly inefficient and difficult to combine with the rest of the algorithm. Indeed, the issue with hashing is not obtaining a masked bit string (which could be done with something like SHA-3), but expanding that bit string into a random-looking polynomial $\mathbf{c}$ of fixed, low Hamming weight in a masked form. The corresponding operation is really hard to write down as a circuit. Moreover, even if that could be done, it would be terrible for performance because subsequent multiplications by $\mathbf{c}$ are no longer products by a known sparse constant, but full-blown ring operations that have to be fully masked.

However, more importantly, this masking *should* intuitively be unnecessary. Indeed, when we see the signature scheme as the conversion of an identification protocol under the Fiat–Shamir transform, the hash function computation corresponds to the verifier's sampling of a random challenge $\mathbf{c}$ after it receives the commitment value $\mathbf{r}$ from the prover. In particular, the verifier always learns the commitment value $\mathbf{r}$ (corresponding to the input of the hash function), so if the identification protocol is "secure", one should always be able to reveal this value without compromising security. However, the security of the signature scheme only offers weak guarantees on the security of the underlying identification protocol, as discussed by Abdalla et al. [Abd+02].

In usual Fiat–Shamir signatures, this is never an issue because the commitment value can always be publicly derived from the signature (as it is necessary for signature verification). However, things are more subtle in the Fiat–Shamir with aborts paradigm, since the value $\mathbf{r}$ is not normally revealed in executions of the signing algorithm that do not pass the rejection sampling step. In our setting, though, we would like to unmask the value to compute the hash function in all cases, before knowing whether the rejection sampling step will be successful. If we do so, the side-channel attacker can learn the pair $(\mathbf{r}, \mathbf{c})$ corresponding to rejected executions as well, and the original security proof does not cover this, nor does security with this additional leakage look reducible to the original security assumption. The purpose of the next paragraphs is to prove that unmasking the value $\mathbf{r}$ does not affect the security of the scheme.

This stronger security requirement can be modeled as the unforgeability under chosen message attacks of a modified version of the GLP signature scheme in which the pair $(\mathbf{r}, \mathbf{c})$ is made public when a rejection occurs. We call this modified scheme $\mathbf{r}$-GLP, and describe it as Algorithm 18. The modification means that, in the EUF-CMA security game, the adversary gets access not only to correctly generated GLP signatures but also to pairs $(\mathbf{r}, \mathbf{c})$ when rejection occurs, which is exactly the setting that arises as a result of unmasking the value $\mathbf{r}$.

We now introduce a new computational problem to handle the security of our modified $\mathbf{r}$-GLP scheme.

Hard Problem 9 — Rejected-Decisional Compact Knapsack problem (R-DCK)

Let $q, n, \mathtt{Q}_\mathrm{s}, \alpha, \kappa$ be integer parameters.

Given $(\mathbf{a}, \mathbf{b}_i, \mathbf{c}_i) \in R_q \times R_q \times \mathcal{D}_\alpha^n$ for $i \in [0, \mathtt{Q}_\mathrm{s} - 1]$

Distinguish whether this sample has been drawn from

1. the procedure where
   - $\mathbf{a}$ is drawn uniformly in R_q;
   - the $\mathbf{b}_i$ are drawn uniformly in R_q;
   - the $\mathbf{c}_i$ are drawn uniformly in $\mathcal{D}_\alpha^n$.
2. or the procedure where
   - elements $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{a})$ are firstly drawn uniformly in $R_1^2 \times R_q$;
   - for each i, elements $(\mathbf{c}_i, \mathbf{y}_{1,i}, \mathbf{y}_{2,i})$ are drawn uniformly in $\mathcal{D}_\alpha^n \times R_{q,\kappa}^2$. This step is repeated until $\mathbf{s}_1\mathbf{c}_i + \mathbf{y}_{1,i} \notin R_{q,\kappa-\alpha}$ or $\mathbf{s}_2\mathbf{c}_i + \mathbf{y}_{2,i} \notin R_{q,\kappa-\alpha}$.
   - the final samples are $(\mathbf{a}, \mathbf{b}_i, \mathbf{c}_i) = (\mathbf{a}, \mathbf{a}\mathbf{y}_{1,i} + \mathbf{y}_{2,i}, \mathbf{c}_i)$.

To obtain a scheme that more directly follows the original one and to keep the overhead reasonable, we propose to use $\mathbf{R\text{-}DCK}_{q,n,\mathtt{Q}_\mathrm{s},\alpha,\kappa}$ as an extra assumption, which we view as a pragmatic compromise. The assumption is admittedly somewhat artificial, but the same can be said to begin with, of $\mathbf{DCK}_{q,n}$ itself. Heuristically, $\mathbf{R\text{-}DCK}_{q,n,\mathtt{Q}_\mathrm{s},\alpha,\kappa}$ is similar, except that it removes smaller (hence "easier to distinguish") instances from the distribution. In essence, one expects that this makes distinguishing harder, even though one cannot really write down a reduction to formalize that intuition.

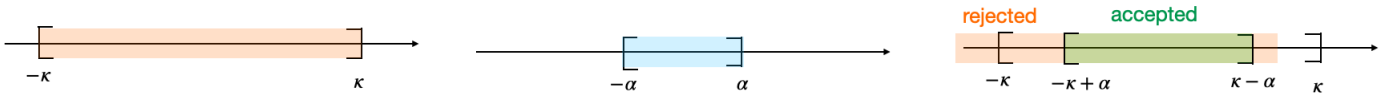


Figure 3.6: *Graphical intuition for computing* $P\left[\|\mathbf{s}_1\mathbf{c}+\mathbf{y}_1\|_\infty > \kappa-\alpha\right] = P\left[\|\mathbf{s}_2\mathbf{c}+\mathbf{y}_2\|_\infty\right]$. *Without loss of generality, we only consider* $\mathbf{y}_1$, $\mathbf{s}_1$, $\mathbf{z}_1$. *To the left, we represent the distribution for the coefficients of* $\mathbf{y}_1$. *In the middle, we represent the possible values for the coefficients of* $\mathbf{s}_1\mathbf{c}$. *To the right, the distribution of* $\mathbf{y}_i$ *is shifted by* $\mathbf{s}_i\mathbf{c}$ *to obtain the distribution of* $\mathbf{z}_1$. *The accepted (resp. rejected)* $\mathbf{z}_1$ *are in green (resp. in orange.). We see that, no matter the shift of* $\mathbf{s}_i\mathbf{c}$, *the fraction of rejected coefficients will stay* $= \frac{2\alpha}{2\kappa+1}$.

Algorithm 18 — Tweaked signature with public $\mathbf{r}$

Data: m, $pk = (\mathbf{a}, \mathbf{t})$, $sk = (\mathbf{s_1}, \mathbf{s_2})$
Result: Signature sig

$\mathbf{y_1} \xleftarrow{\$} R_{q,\kappa}$
$\mathbf{y_2} \xleftarrow{\$} R_{q,\kappa}$
$\mathbf{r} \coloneqq \mathbf{a}\mathbf{y_1} + \mathbf{y_2}$
$\mathbf{c} \coloneqq \mathsf{hash}(\mathbf{r}, m)$
$\mathbf{z_1} \coloneqq \mathbf{s_1}\mathbf{c} + \mathbf{y_1}$
$\mathbf{z_2} \coloneqq \mathbf{s_2}\mathbf{c} + \mathbf{y_2}$
if $\mathbf{z_1}$ *or* $\mathbf{z_2} \notin R_{q,\kappa-\alpha}$ **then**
  $(\mathbf{z_1}, \mathbf{z_2}) \coloneqq (\perp, \perp)$
end
return sig $\coloneqq (\mathbf{z_1}, \mathbf{z_2}, \mathbf{c}, \mathbf{r})$

The following theorem states that the modified scheme is indeed secure, at least if we are willing to assume the hardness of the additional $\mathbf{R\text{-}DCK}_{q,n,\mathbb{Q}_s,\alpha,\kappa}$ assumption. We refer to the original paper [Bar+18] for the game-based proof.

Theorem 5. *Assuming the hardness of the* $\boldsymbol{DCK}_{q,n}$ *and* $\boldsymbol{R\text{-}DCK}_{q,n,\mathbb{Q}_s,\alpha,\kappa}$ *problems, the signature* $\mathbf{r}$*-GLP, presented in Algorithm 18 is EUF-CMA secure in the random oracle model.*

Remark 8. *One can think that Algorithm 18 is insecure*[3] *because the adversary have access to the rejected* $\mathbf{c}$*'s and the rejection probability is accessible (by counting the number of rejected signatures over the total number of signature queries). However, the rejection probability is, by construction, independent from* $\mathbf{s}_1$, $\mathbf{s}_2$ *and* $\mathbf{c}$. *Thus, even conditioned with smartly chosen* $\mathbf{c}$, *the adversary cannot gain any information from this probability. More precisely, the former probability is*

$$P\left[\|\mathbf{s}_1\mathbf{c}+\mathbf{y}_1\|_\infty > \kappa-\alpha\right] = P\left[\|\mathbf{s}_2\mathbf{c}+\mathbf{y}_2\|_\infty > \kappa-\alpha\right] = \frac{2\alpha}{2\kappa+1}.$$

It can be seen graphically with Fig. 3.6.

Remark 9. *We can avoid the non-standard assumption* $\boldsymbol{R\text{-}DCK}_{q,n,\mathbb{Q}_s,\alpha,\kappa}$ *by hashing not* $\mathbf{r}$ *but* $f(\mathbf{r})$ *for some statistically hiding commitment* f *(which can itself be constructed under* $\boldsymbol{DCK}_{q,n}$, *or standard lattice assumptions). We refer to the appendices of [Bar+18] for details. The downside of that approach is that it incurs a non-negligible overhead in terms of key size, signature size, and to a lesser extent signature generation time.*

[3] We thank Damien Stehlé for suggesting this approach.

3.3.3 Masking r-GLP signature scheme

The whole **r**-GLP scheme is turned into a functionally equivalent scheme secure in the d-probing model with public outputs. Note that it suffices to mask the key generation in the d-probing model and the signature in the d-probing model with public output $\mathbf{r}$, since the verification step does not manipulate sensitive data.

Remark 10. *As the number of signature queries per private key can be high (up to* 2^{64} *as required by the NIST competition), whereas, the key generation algorithm is typically only executed once per private key, the vulnerability of the key generation to side-channel attacks is, therefore, less critical. Only possible attacks are the side-channel attacks with single trace (see the Section 1.1.3 in the introductory chapter). However, here, the key generation can be easily provably masked, so we still present the technique.*

For simplicity, we will show the masking on a single iteration version of the signature. The masking can be generalized by calling the masked signature again with an SNI refreshing of the private key $(\mathbf{s}_1, \mathbf{s}_2)$ if it fails.

To ensure protection against d-th order attacks, we suggest a masking countermeasure with $d+1$ shares for the following sensitive data : $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{s}_1$ and $\mathbf{s}_2$. All the public variables are $(\mathbf{a}, \mathbf{t})$ (i.e., the public key), m (i.e., the message), rs (i.e., the bit corresponding to the success of the rejection sampling), $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ (i.e., the signature). As mentioned before, because of the need of $\mathbf{r}$ recombination, even if $\mathbf{r}$ is an intermediate value, it is considered as a public output. Most operations carried out in the **r**-GLP signing algorithm are arithmetic operations modulo q, so we would like to use an arithmetic masking. The issue is that at some points of the algorithm, we need to perform operations that are better expressed using Boolean masking. Those parts will be extracted from both the key generation and the signature to be protected individually and then securely composed.

We present the masked key generation in Algorithm 19 and the masked signature algorithm in Algorithm 20.

Algorithm 19 — Masked **r**-GLP key generation

Result: Signing key sk, verification key pk

$(\mathbf{s}_{1,i})_{0\leq i\leq d} \leftarrow \mathsf{UGen}(1)$
$(\mathbf{s}_{2,i})_{0\leq i\leq d} \leftarrow \mathsf{UGen}(1)$
$\mathbf{a} \xleftarrow{\$} R_q$
$(\mathbf{t}_i)_{0\leq i\leq d} := \mathsf{MAdd}(\mathbf{a}, (\mathbf{s}_{1,i})_{0\leq i\leq d}, (\mathbf{s}_{2,i})_{0\leq i\leq d})$
$\mathbf{t} := \mathsf{Full}+_q ((\mathbf{t}_i)_{0\leq i\leq d})$
$sk := ((\mathbf{s}_{1,i})_{0\leq i\leq d}, (\mathbf{s}_{2,i})_{0\leq i\leq d})$
$pk := (\mathbf{a}, \mathbf{t})$

Algorithm 20 — Masked r-GLP signature

Data: m, $pk = (\mathbf{a}, \mathbf{t})$, $sk = ((\mathbf{s}_{1,i})_{0\leq i\leq d}, (\mathbf{s}_{2,i})_{0\leq i\leq d})$
Result: Signature sig

1 $(\mathbf{y}_{1,i})_{0\leq i\leq d} \leftarrow \mathsf{UGen}(\kappa)$
2 $(\mathbf{y}_{2,i})_{0\leq i\leq d} \leftarrow \mathsf{UGen}(\kappa)$
3 $(\mathbf{r}_i)_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{a}, (\mathbf{y}_{1,i})_{0\leq i\leq d}, (\mathbf{y}_{2,i})_{0\leq i\leq d})$
4 $\mathbf{r} \coloneqq \mathsf{Full} +_q ((\mathbf{r}_i)_{0\leq i\leq d})$
5 $\mathbf{c} \coloneqq \mathsf{hash}(\mathbf{r}, m)$
6 $(\mathbf{z}_{1,i})_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{c}, (\mathbf{s}_{1,i})_{0\leq i\leq d}, (\mathbf{y}_{1,i})_{0\leq i\leq d})$
7 $(\mathbf{z}_{2,i})_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{c}, (\mathbf{s}_{2,i})_{0\leq i\leq d}, (\mathbf{y}_{2,i})_{0\leq i\leq d})$
8 $rs \coloneqq \mathsf{RS}((\mathbf{z}_{1,i})_{0\leq i\leq d}, \kappa - \alpha) \wedge \mathsf{RS}((\mathbf{z}_{2,i})_{0\leq i\leq d}, \kappa - \alpha)$
9 **if** $rs = 1$ **then**
10 $\quad (\mathbf{z_1}, \mathbf{z_2}) \coloneqq (\mathsf{Full} +_q ((\mathbf{z}_{1,i})_{0\leq i\leq d}), \mathsf{Full} +_q ((\mathbf{z}_{2,i})_{0\leq i\leq d}))$
11 **else**
12 $\quad (\mathbf{z_1}, \mathbf{z_2}) \coloneqq (\perp, \perp)$
13 **end**
14 **return** sig $\coloneqq (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{r})$

Theorem 6. *The masked* **r***-GLP sign in Algorithm 20 is d-*`MIo`* secure with public output* $\{b, rs\}$[4].

Remark 11 (Public outputs b and rs)**.** *Recall that b is a public output of* UGen *(Gadget 10). Its value corresponds to the conditions of rejection in the uniform data generation (or equivalently the number of iterations of* UGen *(Gadget 10)). Giving it to the attacker does not impact the security of the scheme. Besides, the value rs, defined in line 9 in Algorithm 20, corresponds to the condition of rejection (or equivalently the number of iterations of Algorithm 20). Such knowledge does not impact on the scheme's security because the rejection probability does not depend on the position of the coefficients.*

Proof: We refer to Table 3.1 for the security properties of the gadgets. Let us assume that an attacker has access to $\delta \leq d$ observations on the whole signature scheme. Then, we want to prove that all these δ observations can be perfectly simulated with at most δ shares of each secret among $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{s}_1$ and $\mathbf{s}_2$ and the public variables. With

[4]In **r**-GLP, **r** is returned along with the signature, so it is not considered as a public output. Besides, recall that b is a public output of UGen (Gadget 10), it corresponds to the number of iterations of the generation before outputting a sample.

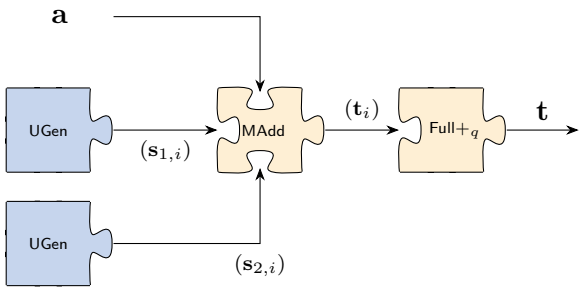


Figure 3.7: *Composition of GLP masked key generation*

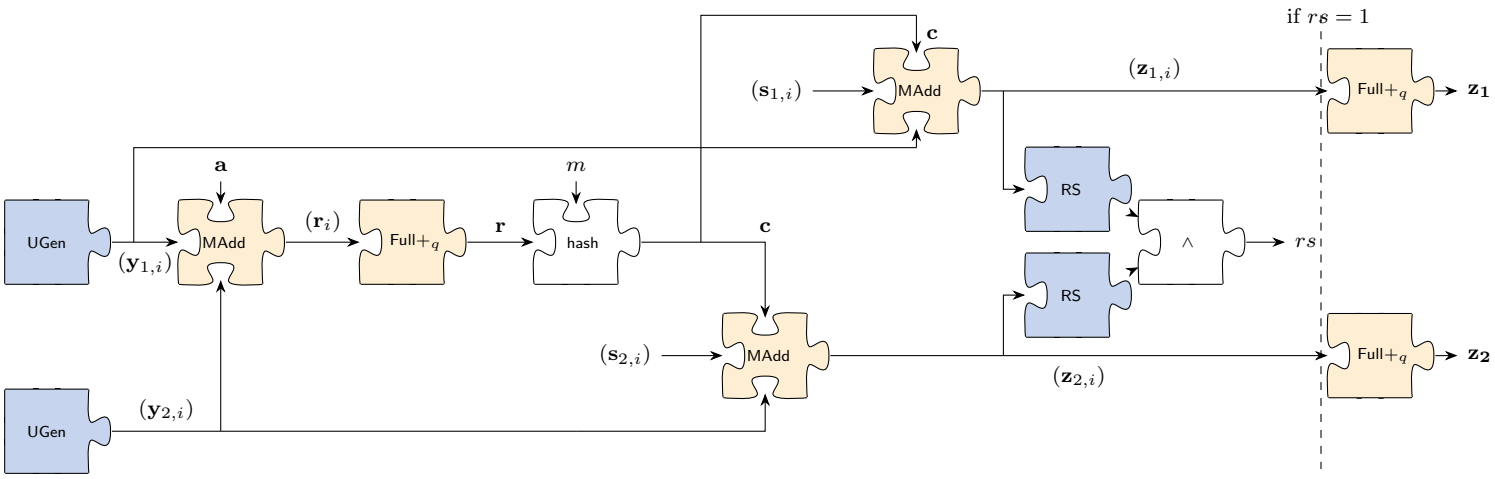


Figure 3.8: *Composition of GLP Signature*

such a result, the signature scheme is then secure in the d-probing model since no set of at most d observations would give information on the secret values.

In the following, we consider the following distribution of the attacker's δ observations: δ_1 (resp. δ_2) on the instance of UGen that produces shares of $\mathbf{y}_1$ (resp. $\mathbf{y}_2$), δ_3 on MAdd, δ_4 on Full+$_q$ of $\mathbf{r}$, δ_5 (resp. δ_6) on MAdd which produces $\mathbf{z}_1$ (resp. $\mathbf{z}_2$), δ_7 (resp. δ_8) on the instance of RS for $\mathbf{z}_1$ (resp. $\mathbf{z}_2$) and δ_9 (resp. δ_{10}) on Full+$_q$ of $\mathbf{z}_1$ (resp. $\mathbf{z}_2$). Some other observations can be made on the hash and $\wedge$, their number will not matter during the proof. Finally, we have $\sum_{i=1}^{10} \delta_i \leq \delta$.

Now, we build the proof from right to left as follows.

Both last Full+$_q$ blocks in the very end of mSign are d-NI secure. Therefore, all the observations performed during the execution of Full+$_q$ on $\mathbf{z}_1$ (resp. $\mathbf{z}_2$) can be perfectly simulated with at most δ_9 (resp. δ_{10}) shares of $\mathbf{z}_1$ (resp. $\mathbf{z}_2$).

RS is d-NIo secure and does not return any sensitive element. Thus, all the observations performed in gadget RS can be perfectly simulated with at most δ_7 shares of $\mathbf{z}_1$ and δ_8 shares of $\mathbf{z}_2$. So, after RS, the observations can be simulated with $\delta_7 + \delta_9$ shares of $\mathbf{z}_1$ and $\delta_8 + \delta_{10}$ shares of $\mathbf{z}_2$.

MAdd is d-NI secure as well; thus, all the observations from the call of MAdd on $\mathbf{y}_1$ can be perfectly simulated with $\delta_5 + \delta_7 + \delta_9 \leq \delta$ shares of $\mathbf{y}_1$ and $\mathbf{s}_1$. Respectively, on $\mathbf{y}_2$, the observations can be perfectly simulated from $\delta_6 + \delta_8 + \delta_{10} \leq \delta$ shares of $\mathbf{y}_2$ and $\mathbf{s}_2$.

The left Full+$_q$ gadget is d-NI secure and does not return any sensitive element, then all the observations performed in this gadget can be perfectly simulated with at most δ_4 shares of $\mathbf{r}$.

The left MAdd gadget is d-NI secure; thus, all the observations from its call can be perfectly simulated with at most $\delta_3 + \delta_4$ shares of each one of the inputs $\mathbf{y}_1$ and $\mathbf{y}_2$.

UGen is also d-NI secure; thus, we need to ensure that the number of reported observations does not exceed δ. At the end of UGen, the simulation relies on $(\delta_3 + \delta_4) + (\delta_5 + \delta_7 + \delta_9) \leq \delta$ shares of $\mathbf{y}_1$ and $(\delta_3 + \delta_4) + (\delta_6 + \delta_8 + \delta_{10}) \leq \delta$ shares of $\mathbf{y}_2$. With the additional δ_1 (resp. δ_2) observations performed on the first (resp. the second) instance of UGen, the number of observations remains below δ, which is sufficient to ensure the security of the whole scheme in the d-probing model. □

Theorem 7. *The masked GLP key generation in Algorithm 19 is d-NIo secure with public output b.*

Proof: We still refer to Table 3.1 for the security properties of the gadgets.

Here too, let us assume that an attacker has access to $\delta \leq d$ observations on the whole signature scheme. Hence, we want to prove that all these δ observations can be perfectly simulated with at most δ shares of each secret among $\mathbf{s}_1$ and $\mathbf{s}_2$.

We now consider the following distribution of the attacker's δ observations: δ_1 (resp. δ_2) on the instance of UGen that produces shares of $\mathbf{s}_1$ (resp. $\mathbf{s}_2$), δ_3 on MAdd, and δ_4 on Full$+_q$, such that $\sum_{i=1}^{4} \delta_i = \delta$.

Now, we build the proof from right to left: Full$+_q$ is used at the very end of the key generation, and it is d-NI secure. Thus, all the observations from the call of Full$+_q$ can be perfectly simulated with $\delta_4 \leq \delta$ sensitive shares of the input $\mathbf{t}$.

MAdd is d-NI; thus, all the observations from its call can be perfectly simulated with at most $\delta_3 + \delta_4 \leq \delta$ shares of each one of the inputs $\mathbf{s}_1$ and $\mathbf{s}_2$.

UGen is d-NIo, thus, we need to ensure that the number of reported observations does not exceed δ. At the end of UGen, the simulation relies on $(\delta_3 + \delta_4) \leq \delta$ shares of $\mathbf{s}_1$ and $\mathbf{s}_2$. With the additional δ_1 (resp. δ_2) observations performed on the first (resp. the second) instance of UGen, the number of observations on each block remains below δ. All the observations can thus be perfectly simulated with the only knowledge of the outputs; that is, the key generation algorithm is this d-NIo secure.

□

3.3.4 EUF-CMA in the d-probing model

Let UnmaskedRefresh be the unmasked version of a refreshing gadget. This algorithm *is not* subject to probing as in Security Model 3; the attacker cannot probe during the key update. Its output, however, is given to the signature oracle and can be probed.

Algorithm 21 — UnmaskedRefresh

Data: $(x_i)_{0 \leq i \leq d}$
for $0 \leq i \leq d-1$ **do**
  Draw a uniformly random x'_i with the same bit size as x_i
end
$x'_d \coloneqq \sum_{k=0}^{d} x_k - \sum_{k=0}^{d-1} x'_k$
return $(x_i)_{0 \leq i \leq d}$

Theorem 8. *Assuming the hardness of the* $\boldsymbol{DCK}_{q,n}$ *and* $\boldsymbol{R\text{-}DCK}_{q,n,\mathtt{Q_s},\alpha,\kappa}$ *problems, the signature scheme* $\mathbf{r}$*-GLP masked at order d with key update algorithm* UnmaskedRefresh *is EUF-CMA secure in the d-probing model (see Security Model 3) and in the random oracle model.*

Proof: Let $\mathrm{Adv}^{d\text{-probing-EUF-CMA}}$ be the advantage of an adversary $\mathcal{A}$ against the d-probing-EUF-CMA security game for $\mathbf{r}$-GLP masked at order d with key update algorithm UnmaskedRefresh. Remark that b is drawn under a publicly known distribution and is independent of the secret. Hence the advantage of an adversary against the EUF-CMA security game for GLP with returned value b, rs is exactly $\mathrm{Adv}^{\text{EUF-CMA}}$,

the advantage of an adversary against the EUF-CMA security game for **r**-GLP. This advantage is negligible and has been quantified in the proof of Theorem 5 (we refer to [Bar+18] for the details).

In the following, we will prove that $\mathrm{Adv}^{d\text{-probing-EUF-CMA}} = \mathrm{Adv}^{\text{EUF-CMA}}$, which is sufficient to conclude the proof. Indeed, thanks to Theorem 5, under the hardness of the $\mathbf{DCK}_{q,n}$ and $\mathbf{R\text{-}DCK}_{q,n,\mathtt{Q_s},\alpha,\kappa}$, $\mathrm{Adv}^{\text{EUF-CMA}}$ and so $\mathrm{Adv}^{d\text{-probing-EUF-CMA}}$ will be negligible.

In the d-probing-EUF-CMA setting, the signature oracle called S is stateful. Let us define this oracle by the following:

$$S(m, \mathscr{O}_{\mathsf{Sign}}) = \left(\mathsf{Sign}(m, pk, sk^{(i-1)}), \mathscr{L}_{\mathsf{Sign}}\right)$$

where $i \in [1, \mathtt{Q}]$ is the state, i.e. the index of the query, and $sk^{(i)} = \mathsf{KeyUpdate}^i(sk)$ (with the convention $sk^0 = sk$).

Leakage Simulators

- According to Theorem 6, **r**-GLP Sign is $d-\mathtt{NIo}$ secure. One subtlety is that the attacker $\mathcal{A}$ is able to make d observations on each call to the signature oracle but not during the key refreshings.
  For any set of at most d observations during the i^{th} call to the signature oracle, the $d-$NI security ensures the existence of a perfectly simulated leakage $\mathscr{L}_{\mathsf{Sign}}^{(i),\mathrm{Sim}}$ that is generated from at most d shares of $sk^{(i-1)}$. Because the attacker is not able to get any observation during the UnmaskedRefresh, if $i \neq 1$, the d shares of $sk^{(i-1)}$ can be perfectly simulated from random. When $i = 1$, the observations are directly simulated with at most d shares of $sk^{(0)} = sk$. In total, the $\mathtt{Q}$ leakages $\left(\mathscr{L}_{\mathsf{Sign}}^{(1),\mathrm{Sim}} ... \mathscr{L}_{\mathsf{Sign}}^{(\mathtt{Q}),\mathrm{Sim}}\right)$ are perfectly simulated from at most d shares of sk. Thus, they are independent of the unmasked value of the secret.

- Identically, according to Theorem 7, for any set of at most d observations during the key generation, there exists a perfectly simulated leakage $\mathscr{L}_{\mathsf{KeyGen}}^{\mathrm{Sim}}$ that is independent of the unmasked value of the secret.

Here are the hybrid games involved in the security proof.

Game $\mathbf{G}_0$: This game is the security game of the EUF-CMA security in the d-probing model.

1. $\mathscr{O}_{\mathsf{KeyGen}} \leftarrow \mathcal{A}$
2. $\left((sk, pk), \mathscr{L}_{\mathsf{KeyGen}}\right) \leftarrow \mathsf{ExecObs}(\mathscr{O}_{\mathsf{KeyGen}}, \mathsf{KeyGen}, 1^\lambda)$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{H,S}(pk, \mathscr{L}_{\mathsf{KeyGen}})$
4. **return** 1 if $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$ and $|\mathscr{O}_{\mathsf{KeyGen}}| \leq d$ and $\forall i \in \{1, \ldots, \mathtt{Q}\}$, $|\mathscr{O}_{\mathsf{Sign}}^{(i)}| \leq d$ and (m^*, σ^*) has not been returned by the signature oracle
   return 0 otherwise.

Game $\mathbf{G}_1$: Let S' be a copy of S that outputs $\mathscr{L}_{\mathsf{Sign}}^{(i),\mathrm{Sim}}$ instead of $\mathscr{L}_{\mathsf{Sign}}^{(i)}$ for all i. In the following game, all the leakages given to the attacker are replaced by the simulated leakages.

1. $\mathscr{O}_{\mathsf{KeyGen}} \leftarrow \mathcal{A}$
2. $\big((sk, pk), \mathscr{L}_{\mathsf{KeyGen}}\big) \leftarrow \mathsf{ExecObs}(\mathscr{O}_{\mathsf{KeyGen}}, \mathsf{KeyGen}, 1^\lambda)$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{H,S'}(pk, \mathscr{L}^{\mathrm{Sim}}_{\mathsf{KeyGen}})$
4. **return** 1 if $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$ and $|\mathscr{O}_{\mathsf{KeyGen}}| \leq d$ and $\forall i \in \{1, \ldots, \mathtt{Q}\}$, $|\mathscr{O}^{(i)}_{\mathsf{Sign}}| \leq d$ and (m^*, σ^*) has not been returned by the signature oracle
return 0 otherwise.

By definition of $\mathscr{L}^{\mathrm{Sim}}$, this game is perfectly indistinguisable from $\mathbf{G}_0$.

Game $\mathbf{G}_2$: This game is the security game of the EUF-CMA security.

1. $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{H,\mathsf{Sign}(\cdot,sk)}(pk)$
3. **return** 1 iff $\textsc{Verify}(pk, m^*, \sigma^*) = 1$ and (m^*, σ^*) has not been returned by the oracle $\mathsf{Sign}(\cdot, sk)$.

Since in game $\mathbf{G}_1$, $|\mathscr{O}_{\mathsf{KeyGen}}| \leq d$ and $|\mathscr{O}^{(i)}_{\mathsf{Sign}}| \leq d$ for all $i \in [1, \mathtt{Q}]$, having access to $\mathscr{L}^{\mathrm{Sim}}$ provides no advantage to the attacker. Thus, the advantage for $\mathbf{G}_1$ is the same as the advantage for $\mathbf{G}_2$.

Finally $\mathrm{Adv}^{d\text{-probing-EUF-CMA}} = \mathrm{Adv}^{\text{EUF-CMA}}$. □

3.3.5 Proof of concept and performance

In our team, Mehdi Tibouchi has carried out an implementation of our masking countermeasure based on a recent, public domain implementation of the GLP signature scheme called GLYPH [Cho17]. The GLYPH scheme actually features a revised set of parameters supposedly achieving a greater level of security (namely, $n = 1024$, $q = 59393$, $\kappa = 16383$ and $\alpha = 16$), as well as a modified technique for signature compression. We do not claim to vouch for those changes but stress that, for our purposes, they are essentially irrelevant. Indeed, our countermeasure's overhead only depends on the masking order d, the bit size of the Boolean masks (which should be chosen as 32 bits both for GLYPH and the original GLP parameters) and the degree n of the ring R_q (which is the same in GLYPH as in the high-security GLP parameters). Therefore, our results on GLYPH should carry over to a more straightforward implementation of GLP as well.

Implementation results on a single core of an Intel Core i7-3770 CPU are provided in Table 3.2. In particular, we see that the overhead of our countermeasure with 2, 3 and 4 shares (secure in the d-probing model for $d = 1, 2, 3$ respectively) is around 15×, 30× and 73×. Given the complete lack of optimizations of this implementation, we believe that this proof of concept is quite promising. The memory overhead is linear in the masking order, so quite reasonable in practice (all masked values are simply represented as a vector of shares).

3.3.6 Extension to Crystals-Dilithium signature scheme

Our work on GLP led to a concrete masked implementation of Dilithium with experimental leakage tests [Mig+19]. In the latter, Migliore *et al.* noticed that replacing the prime modulus by a power of two allows obtaining a considerably more efficient masked scheme, by a factor of 7.3 to 9 for the most time-consuming masking operations.

Table 3.2: *Implementation results. Timings are provided for* 100 *executions of the signing algorithm, on one core of an Intel Core i7-3770 CPU-based desktop machine.*

Number of shares $(d+1)$	Unprotected	2	3	4	5
Total CPU time (s)	0.540	8.15	16.4	39.5	62.1
Masking overhead	—	**×15**	**×30**	**×73**	**×115**

3.4 Application to BLISS signature scheme

A full description of BLISS cryptosystem has been presented in the previous chapter in Section 2.4.1. We studied how to mask the signature algorithm of BLISS in addition to protecting it against timing attacks. The key generation of BLISS can be masked using sorting networks and a masked polynomial generation. Nevertheless, the key generation is now obsolete in terms of complexity and performance compared to BLISS-B or more recent signature implementations (Dilithium, qTESLA). Thus, in this thesis, we focus on presenting the masked signature algorithm.

3.4.1 Masking BLISS signature algorithm

Similar to **r**-GLP, let **u**-BLISS be the variant of BLISS that outputs $\mathbf{u}$ even in case of failure. We here justify the security of **u**-BLISS and prove the d-probing security of the key generation and the signature procedures. For the sake of clarity, we focus on a single iteration of the latter. In other words, from now on, the signature algorithm considered is the same as in Algorithm 9 except that if the rejection sampling asks for a restart, the algorithm outputs $\perp$. The masking can be generalized by calling the masked signature algorithm with a refreshed private key when it fails.

For efficiency purposes, our masking countermeasure splits each sensitive data, namely $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{z}_1$, $\mathbf{z}_2$ into $d+1$ shares and the intermediate variables that strictly depend on them. The public variables $(\mathbf{a}_1, \mathbf{a}_2)$ (i.e., the public key), m (i.e., the message), rs (i.e., the bit corresponding to the success of the rejection sampling), $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ (i.e., the signature) are left unmasked. Furthermore, because anyone can recombine $\lfloor \mathbf{u} \rceil_{\mathtt{Mst}} \bmod p$, even if u is an intermediate value, it is considered as a public output along with the bit rs.

We present the masked BLISS signature algorithm in Algorithm 22. The PolyEval() function refers to the masking of the isochronous rejection sampling procedure presented in Algorithm 10 in Chapter 2. It can easily be transformed to ensure d-probing security as follows. Step 1 computes two elements x_1 and x_2 from sensitive values $\mathbf{s}$ and $\mathbf{z}$. Multiplications must be processed with function sec& in the masked version. As for Step 2, two sets of $d+1$ Boolean shares are generated at random in $\{0,1\}$ to represent the secret bits u_1 and u_2. Steps 3 and 4 require the computation of $\exp(x_1)$ and $\cosh(x_2)$ with x_1 and x_2 sensitive values shared in Step 1. Thanks to the polynomial approximation of these two functions, as described in Sections 2.4.3 and 2.4.4, the evaluation of exp and cosh for these two sharings is only a combination of linear squaring and sec& operations. As for their comparison with functions of u_1 and u_2, the computed arithmetic sharings are first converted into Boolean sharings (the sharing of u_1 can be first converted into an arithmetic masking to be subtracted to $\exp(x_1)$ which allows a comparison with public values). Then, a secure comparison is performed between Boolean sharings and outputs

two masked bits of a and b. Finally, the last multiplication in Step 5 is computed with sec&, before a call to Full$+_q$ outputs rs.

Algorithm 22 — Masked $\mathbf{u}$-BLISS signature

Data: m, $pk = (\mathbf{a}, \mathbf{t})$, $sk = ((\mathbf{s}_{1i})_{0\leq i\leq d}, (\mathbf{s}_{2i})_{0\leq i\leq d})$
Result: Signature sig

1 $(\mathbf{y}_{1i})_{0\leq i\leq d} \leftarrow \mathsf{GGen}_\sigma()$
2 $(\mathbf{y}_{2i})_{0\leq i\leq d} \leftarrow \mathsf{GGen}_\sigma()$
3 $(\mathbf{u}_i)_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\zeta \cdot \mathbf{a}_1, (\mathbf{y}_{1i})_{0\leq i\leq d}, (\mathbf{y}_{2i})_{0\leq i\leq d})$
4 $\mathbf{u} \coloneqq \mathsf{Full} +_q ((\mathbf{u}_i)_{0\leq i\leq d})$
5 $\mathbf{c} \coloneqq \mathsf{hash}(\lfloor \mathbf{u} \rceil_d, m)$
6 $(b_i)_{0\leq i\leq d} \xleftarrow{\$} \{0,1\}^{d+1}$
7 $(b_i)_{0\leq i\leq d} \leftarrow \mathsf{B}\rightsquigarrow\mathsf{A}_q((b_i)_{0\leq i\leq d})$
8 $(b_i)_{0\leq i\leq d} \coloneqq (2 \cdot (b_i) - 1 \mod q)_{0\leq i\leq d}$
9 $(\mathbf{z}_{1i})_{0\leq i\leq d} \leftarrow \mathsf{sec\&}((b_i)_{0\leq i\leq d}, (\mathbf{s}_{1i})_{0\leq i\leq d})$
10 $(\mathbf{z}_{2i})_{0\leq i\leq d} \leftarrow \mathsf{sec\&}((b_i)_{0\leq i\leq d}, (\mathbf{s}_{2i})_{0\leq i\leq d})$
11 $(\mathbf{z}_{1i})_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{c}, (\mathbf{z}_{1i})_{0\leq i\leq d}, (\mathbf{y}_{1i})_{0\leq i\leq d})$
12 $(\mathbf{z}_{1i})_{0\leq i\leq d} \leftarrow \mathsf{Ref}_{+_q}((\mathbf{z}_{1i})_{0\leq i\leq d})$
13 $(\mathbf{z}_{2i})_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{c}, (\mathbf{z}_{2i})_{0\leq i\leq d}, (\mathbf{y}_{1i})_{0\leq i\leq d})$
14 $(\mathbf{z}_{2i})_{0\leq i\leq d} \leftarrow \mathsf{Ref}_{+_q}((\mathbf{z}_{2i})_{0\leq i\leq d})$
15 $rs \coloneqq \mathsf{PolyEval}(sk, (\mathbf{z}_{1i})_{0\leq i\leq d}, (\mathbf{z}_{2i})_{0\leq i\leq d}, \mathbf{c})$
16 **if** $rs = 1$ **then**
17 $\quad (\mathbf{z_1}, \mathbf{z_2}) \coloneqq (\mathsf{Full} +_q ((\mathbf{z}_{1,i})_{0\leq i\leq d}), \mathsf{Full} +_q ((\mathbf{z}_{2,i})_{0\leq i\leq d}))$
18 $\quad \mathbf{z}_2^\dagger \coloneqq (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
19 $\quad$ **return** sig $\coloneqq (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{u})$
20 **else**
21 $\quad$ **return** sig $\coloneqq (\perp, \perp, \mathbf{u})$
22 **end**

Theorem 9. *The masked* $\mathbf{u}$*-BLISS sign algorithm is* d*-*`NIo` *secure with public output* rs*.*

Proof: From Table 3.1, all the sub-gadgets involved in the computation of the signature are either d-`NI` secure, d-`NIo` secure, d-`SNI` secure, or they do not manipulate sensitive data. In the d-probing model, we assume that an attacker has access to $\delta \leq d$ exact variables in the whole execution of the signature. Then, we want to prove that all these δ observations can be perfectly simulated with at most δ shares of each secret among y_1, y_2, s_1, and s_2, and the public variables. We consider the following distribution of the attacker's δ observations: δ_1 (resp. δ_2) on the instance of GGen_σ which produces y_1 (resp. y_2), δ_3 on MAdd, δ_4 on the instance of Full$+_q$ following MAdd, δ_5 on hash, δ_6 on B$\rightsquigarrow$A$_q$, δ_7 (resp. δ_8) on the instance of sec& involving s_1 (resp. s_2), δ_9 (resp. δ_{10}) on the instance of MAdd which outputs z_1 (resp. z_2), $\delta_1 1$ (resp. δ_{12}) on the instance of Refresh which outputs z_1 (resp. z_2), δ_{13} on Polyeval, δ_{14} (resp. δ_{15}) on the instance of Full$+_q$ that outputs the unmasked version of z_1 (resp. z_2) such that $\sum_{i=1}^{16} \delta_i = \delta$.

We build the proof from right to left. The final Full$+_q$ are d-`NI` secure. As a consequence, all the observations from their call involving z_1 (resp z_2) can be perfectly simulated with at most $\delta_{14} \leq \delta$ shares of z_1 (resp. at most $\delta_{15} \leq \delta$ shares of z_2). The algorithm referred to as Polyeval is also assumed d-`NI` secure

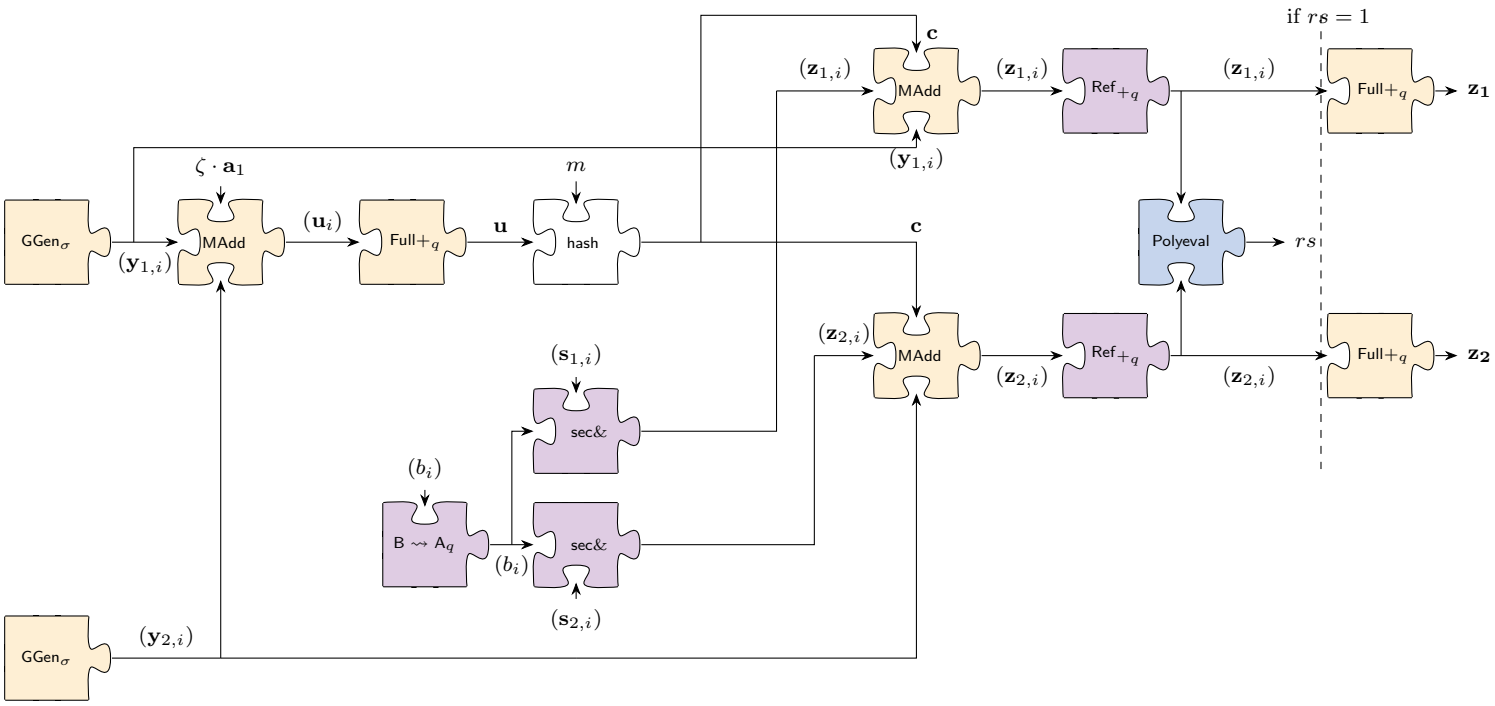


Figure 3.9: *Composition of BLISS Signature*

by construction. Thus, all the observations from its call involving z_1 (resp z_2) can be perfectly simulated with at most $\delta_{13} \leq \delta$ shares of z_1 (resp. at most $\delta_{13} \leq \delta$ shares of z_2) and the bit rs which is public information indicating whether or not the rejection sampling failed. Continuing from right to left, we consider both instances of Refresh. From its d-SNI security property and since the output and local observations are still less than δ, all observations from its call can be perfectly simulated with at most $\delta_{11} \leq \delta$ (resp. $\delta_{12} \leq \delta$) input shares. Both instances of MAdd outputs variables that are immediately refreshed. MAdd is additionally d-NI secure and has δ_{11} (resp. δ_{12}) output observations and δ_9 (resp. δ_{10}) internal ones. In both cases, the addition of the internal and output observations remains below δ. Therefore, the d-NI property makes it possible to simulate all further observations with $\delta_{11} + \delta_9 \leq \delta$ shares of $y_{1,i}$, $s_{1,i}$ and the knowledge of c (resp. $\delta_{12} + \delta_{10} \leq \delta$ shares of $y_{2,i}$, $s_{2,i}$, and the knowledge of c). The sec& instances are d-SNI secure, thus d-NI secure. All the observations after their call can be perfectly simulated with $\delta_7 + \delta_{11} + \delta_9$ shares of $\mathbf{s}_1$ and b (resp. $\delta_8 + \delta_{12} + \delta_{10}$ shares of $\mathbf{s}_2$ and b). The gadget A$_q \rightsquigarrow$ B is d-SNI, the fact that the addition of $(\delta_7 + \delta_{11} + \delta_9) + \delta_8 + \delta_{12} + \delta_{10} \leq \delta$ output observations and δ_6 internal observations is less than δ is enough to guarantee the global security from its location. The hash step only manipulates public data. Full$+_q$ is d-NI secure (its output is given as output of the signature **u**-BLISS, so it is technically not a public output, see Remark 5) and does not return any sensitive variable. Then all the observations performed from this gadget can be perfectly simulated with at most δ_4 shares of u_i. MAdd is d-NI secure. δ_3 observations are performed on its intermediate variables, and at most δ_4 observations are performed on its outputs. As $\delta_3 + \delta_4 \leq \delta$, all further observations can be perfectly simulated with at most $\delta_3 + \delta_4$ shares of y_1, $\delta_3 + \delta_4$ shares of y_2 and the knowledge of the public value $\mathbf{a}_1$. The last step of the proof is to verify that all the internal and output observations on each instance of GGen$_\sigma$ are less than δ. Internal observations are respectively δ_1 and δ_2 while

output observations are bounded by $(\delta_3+\delta_4)+(\delta_{11}+\delta_9)$ and $(\delta_3+\delta_4)+(\delta_{12}+\delta_{10})$ which are both less than δ. The d-NIo property of GGen_σ concludes the proof. □

3.4.2 EUF-CMA in the d-probing model

Based on Section 3.3.4, we can prove that the security of u-BLISS reduces to the EUF-CMA security of the original BLISS scheme by introducing a mild computational assumption which is close to the classical LWE problem (see Theorem 5). This assumption informally states that distinguishing the output distribution of u when a rejection occurs from the uniform distribution on R_{2q} is hard.

Remark 12. *Similarly to Remark 2, assume that an attacker is able to select a subset of signatures by filtering*[5] *the output* $\mathbf{u}$ *(or equivalently* $\mathbf{c}$*) and assume that she has access to the expectation of rejection conditioned with* $\mathbf{c}$*, i.e.* $\mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2,b}[(\mathbf{z}_1,\mathbf{z}_2,\mathbf{c})$ *rejected* $\mid \mathbf{c}]$*. The following computation shows that this quantity is independent from* $\mathbf{s}_1$*,* $\mathbf{s}_2$ *and* $\mathbf{c}$*. Let* $\mathbf{s}_1$*,* $\mathbf{s}_2$ *and* $\mathbf{c}$ *be fixed quantities,*

$$
\begin{aligned}
p_{sc}^{\text{bliss}} &:= \mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2,b}[(\mathbf{z}_1,\mathbf{z}_2,\mathbf{c}) \text{ rejected} \mid \mathbf{c}] = \frac{1}{M}\cdot\mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2,b}\left[\exp\left(-\frac{\|\mathbf{S}\cdot\mathbf{c}\|_2}{2\sigma^2}\right)^{-1}\cosh\left(\frac{\langle\mathbf{z},\mathbf{Sc}\rangle}{\sigma^2}\right)^{-1}\right]\\
&= \frac{1}{M}\cdot\mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2,b}\left[\frac{\rho_\sigma(\|\mathbf{z}\|_2)}{\frac{1}{2}\rho_\sigma(\|\mathbf{z}-\mathbf{Sc}\|_2)+\frac{1}{2}\rho_\sigma(\|\mathbf{z}+\mathbf{Sc}\|_2)}\right]\\
&= \frac{1}{M}\cdot\left(\frac{1}{2}\underbrace{\mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2}\left[\frac{\rho_\sigma(\|\mathbf{y}+\mathbf{Sc}\|_2)}{\frac{1}{2}\rho_\sigma(\|\mathbf{y}\|_2)+\frac{1}{2}\rho_\sigma(\|\mathbf{y}+2\mathbf{Sc}\|_2)}\right]}_{b=0}+\frac{1}{2}\underbrace{\mathbb{E}_{\mathbf{y}_1,\mathbf{y}_2}\left[\frac{\rho_\sigma(\|\mathbf{y}-\mathbf{Sc}\|_2)}{\frac{1}{2}\rho_\sigma(\|\mathbf{y}\|_2)+\frac{1}{2}\rho_\sigma(\|\mathbf{y}-2\mathbf{Sc}\|_2)}\right]}_{b=1}\right)\\
&= \frac{1}{M\rho_\sigma(\mathbb{Z}^m)}\cdot\left(\sum_{\mathbf{y}}\frac{1}{2}\frac{\rho_\sigma(\|\mathbf{y}\|_2)\rho_\sigma(\|\mathbf{y}+\mathbf{Sc}\|_2)}{\frac{1}{2}\rho_\sigma(\|\mathbf{y}\|_2)+\frac{1}{2}\rho_\sigma(\|\mathbf{y}+2\mathbf{Sc}\|_2)}+\sum_{\mathbf{y}}\frac{1}{2}\frac{\rho_\sigma(\|\mathbf{y}\|_2)\rho_\sigma(\|\mathbf{y}-\mathbf{Sc}\|_2)}{\frac{1}{2}\rho_\sigma(\|\mathbf{y}\|_2)+\frac{1}{2}\rho_\sigma(\|\mathbf{y}-2\mathbf{Sc}\|_2)}\right)\\
&= \frac{1}{M\rho_\sigma(\mathbb{Z}^m)}\cdot\left(\sum_{\mathbf{z}_1=\mathbf{y}+\mathbf{Sc}}\frac{\rho_\sigma(\|\mathbf{z}_1-\mathbf{Sc}\|_2)\rho_\sigma(\|\mathbf{z}_1\|_2)}{\rho_\sigma(\|\mathbf{z}_1-\mathbf{Sc}\|_2)+\rho_\sigma(\|\mathbf{z}_1+\mathbf{Sc}\|_2)}+\right.\\
&\qquad\left.\sum_{\mathbf{z}_2=\mathbf{y}-\mathbf{Sc}}\frac{\rho_\sigma(\|\mathbf{z}_2+\mathbf{Sc}\|_2)\rho_\sigma(\|\mathbf{z}_2\|_2)}{\rho_\sigma(\|\mathbf{z}_2+\mathbf{Sc}\|_2)+\rho_\sigma(\|\mathbf{z}_2-\mathbf{Sc}\|_2)}\right).
\end{aligned}
$$

The summation domain for $\mathbf{z}_1$ *and* $\mathbf{z}_2$ *is the same, so,*

$$
\begin{aligned}
p_{sc}^{\text{bliss}} &= \frac{1}{M\rho_\sigma(\mathbb{Z}^m)}\cdot\sum_{\mathbf{z}}\left(\frac{\rho_\sigma(\|\mathbf{z}-\mathbf{Sc}\|_2)\rho_\sigma(\|\mathbf{z}\|_2)}{\rho_\sigma(\|\mathbf{z}-\mathbf{Sc}\|_2)+\rho_\sigma(\|\mathbf{z}+\mathbf{Sc}\|_2)}+\frac{\rho_\sigma(\|\mathbf{z}+\mathbf{Sc}\|_2)\rho_\sigma(\|\mathbf{z}\|_2)}{\rho_\sigma(\|\mathbf{z}+\mathbf{Sc}\|_2)+\rho_\sigma(\|\mathbf{z}-\mathbf{Sc}\|_2)}\right)\\
&= \frac{1}{M\rho_\sigma(\mathbb{Z}^m)}\cdot\sum_{\mathbf{z}}\rho_\sigma(\|\mathbf{z}\|_2) = \frac{1}{M}.
\end{aligned}
$$

3.5 Application to qTesla signature scheme

While the study of GLP and BLISS were proofs of concept, we now focus on applying an optimized masking countermeasure to qTESLA, a candidate to the NIST standardization

[5] We thank Damien Stehlé for suggesting this approach.

Table 3.3: *Parameters for `qTESLA-I` and `qTESLA-III`*

Parameters	`qTESLA-I`	`qTESLA-III`	Description
n	512	1024	Dimension of the ring
q	$4\,205\,569 \approx 2^{22}$	$8\,404\,993 \approx 2^{23}$	Modulus
σ	22.93	10.2	Standard deviation
h	30	48	Nonzero entries of $\mathbf{c}$
E	1586	1147	Rejection parameter
S	1586	1233	Rejection parameter
B	$2^{20}-1$	$2^{21}-1$	Bound for $\mathbf{y}$
`Tail`	21	22	Bits dropped in $[\cdot]_M$

process [Bin+17]. The qTESLA signature scheme is also a Fiat-Shamir lattice-based signature derived from the original work of Lyubashevsky [Lyu12]. This signature is, with Dilithium [Duc+18], one of the most recent iterations of this line of research. We slightly modify the signature and parameters to ease the addition of the countermeasure while keeping the original security. This work has been done with François Gérard [GR19].

3.5.1 The qTESLA signature Scheme

Let us now describe qTESLA [Bin+19], a (family of) lattice-based signatures based on the Ring-LWE problem (see Hard Problem 5) and a round-2 candidate for the NIST post-quantum competition. The signature stems from several iterations of improvements over the original scheme of Lyubashevsky [Lyu12]. It is, in fact, a concrete instantiation of the scheme of Bai and Galbraith [BG14] over ideal lattices. Its direct contender in the competition is Dilithium [Duc+18], which is also based on this same idea of having a lattice variant of Schnorr signature. The security of Dilithium relies on problems over module lattices instead of ideal lattices, in the hope of increasing security by reducing algebraic structure, at the cost of a slight performance penalty.

Hereunder, we explicitly describe the main algorithms, namely key generation, signature and verification. Beforehand, let us briefly recall the functionality of each of the subroutines for completeness. We redirect the interested reader to [Bin+19] or the NIST submission for a detailed description. In Table 3.3, we present the different parameters of qTESLA. We write $x \bmod^{\pm} q$ to denote the unique integer $x_{ct} \in (-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor]$ (where the lower bound is included if q is odd) such that $x_{ct} \equiv x \pmod{q}$. We call this integer the *centered representative* of x modulo q. In the following, PRF is a pseudo-random function, GenA generates a uniformly random polynomial, GaussSampler samples a polynomial according to a Gaussian distribution, CheckS and CheckE verifies that a secret polynomial does not have too large coefficients, ySampler samples a uniformly random polynomial $\mathbf{y} \in R_{q,B}$ (as defined in Definition 13), hash is a collision-resistant hash function, and Enc encodes a bit string into a sparse polynomial $\mathbf{c} \in R_{q,1}$ with $||\mathbf{c}||_1 = h$.

Remark 13. *For the sake of practicability, we focus on the heuristic version of qTESLA as the use case. More specifically, in Section 5.5, we implement our countermeasure in `qTESLA-I` and `qTESLA-III` even though the techniques we are using are not specific to any parameter set. After this work was actually accepted at the CARDIS conference, the heuristic parameter sets, on which our experiments are based, were removed by the qTESLA team. We emphasize that the parameters we use were* not *broken but are not part of the standardization process anymore.*

Algorithm 23 — qTESLA key generation

Result: Signing key $sk = (\mathbf{s}, \mathbf{e}, \text{seed}_a, \text{seed}_y)$, verification key $pk = (seed_a, \mathbf{t})$

counter $\coloneqq 1$
pre-seed $\xleftarrow{\$} \{0,1\}^\kappa$
$\text{seed}_{s,e,a,y} \coloneqq \mathsf{PRF}(\text{pre-seed})$
$\mathbf{a} \coloneqq \mathsf{GenA}(\text{seed}_a)$
repeat
  $\mathbf{s} \coloneqq \mathsf{GaussSampler}(\text{seed}_s, \text{counter})$
  counter $\coloneqq$ counter $+ 1$
until $\mathit{CheckS}(\mathbf{s}) = 0$
repeat
  $\mathbf{e} \coloneqq \mathsf{GaussSampler}(\text{seed}_e, \text{counter})$
  counter $\coloneqq$ counter $+ 1$
until $\mathit{CheckE}(\mathbf{e}) = 0$
$\mathbf{t} \coloneqq \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q$
$sk \coloneqq (\mathbf{s}, \mathbf{e}, \text{seed}_a, \text{seed}_y)$
$pk \coloneqq (seed_a, \mathbf{t})$
return (pk, sk)

Algorithm 24 — qTESLA signature algorithm

Data: message m, $sk = (\mathbf{s}, \mathbf{e}, \text{seed}_a, \text{seed}_y)$
Result: $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$

counter $\coloneqq 1$
$r \xleftarrow{\$} \{0,1\}^\kappa$
$\text{rand} \coloneqq \mathsf{PRF}(\text{seed}_y, r, \mathsf{hash}(m))$
$\mathbf{y} \coloneqq \mathsf{ySampler}(\text{rand}, \text{counter})$
$\mathbf{a} \coloneqq \mathsf{GenA}(\text{seed}_a)$
$\mathbf{v} \coloneqq \mathbf{a} \cdot \mathbf{y} \bmod^{\pm} q$
$\mathbf{c} \coloneqq \mathsf{Enc}(\mathsf{hash}(\lfloor \mathbf{v} \rceil_{\mathtt{Mst}}, m))$
$\mathbf{z} \coloneqq \mathbf{y} + \mathbf{s} \cdot \mathbf{c}$
if $\mathbf{z} \notin R_{q,B-S}$ **then**
  counter $\coloneqq$ counter $+ 1$
  Goto 3
end
return $(\mathbf{z}, \mathbf{c})$

Algorithm 25 — qTESLA verification algorithm

Data: message m, signature $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$ and public key $pk = (seed_a, \mathbf{t})$
Result: 0 if the signature is accepted else -1

$\mathbf{a} \coloneqq \mathsf{GenA}(\text{seed}_a)$
$\mathbf{w} \coloneqq \mathbf{a} \cdot \mathbf{z} - \mathbf{t} \cdot \mathbf{c} \bmod^{\pm} q$
if $\mathbf{z} \notin R_{q,B-S}$ ***or*** $\mathbf{c} \neq \mathsf{Enc}(\mathsf{hash}(\lfloor \mathbf{w} \rceil_{\mathtt{Mst}}, m))$ **then**
  return -1
end
return 0

Key generation (Algorithm 23). The key generation will output an RLWE sample together with some seeds used to generate public parameters and to add a deterministic component to the signing procedure. The algorithm starts by expanding some randomness into a collection of seeds and generates the public polynomial $\mathbf{a}$ before moving on to the two secret values $\mathbf{s}$ and $\mathbf{e}$. Those two values are sampled from a Gaussian distribution and have to pass some checks to ensure that the products $\mathbf{s} \cdot \mathbf{c}$ and $\mathbf{e} \cdot \mathbf{c}$ do not have too large coefficients. After that, the main component $\mathbf{t}$ of the public key is computed as $\mathbf{t} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$. The output consists of the private key $sk = (\mathbf{s}, \mathbf{e}, \text{seed}_a, \text{seed}_y)$ and the public key $pk = (seed_a, \mathbf{t})$.

Sign (Algorithm 24). The sign procedure takes as input a message m and the private key sk, and it outputs a signature $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$. First, in order to generate the randomness needed in the algorithm, a seed is derived from a fresh random value r, seed_y and m. Next, a polynomial $\mathbf{y} \in R_{q,B}$ is sampled to compute the value $\mathbf{v} = \mathbf{a} \cdot \mathbf{y} \bmod^{\pm} q$. The algorithm will now hash the rounded version of $\mathbf{v}$ together with the message and encode the result in a sparse polynomial $\mathbf{c}$ with only h entries in $\{-1, 1\}$. The candidate signature is computed as $\mathbf{z} = \mathbf{y} + \mathbf{s} \cdot \mathbf{c}$. Before outputting the result, two additional checks must be performed: we must ensure that $\mathbf{z}$ is in $R_{q,B-S}$ and that $\mathbf{w} = \mathbf{v} - \mathbf{e} \cdot \mathbf{c} \bmod^{\pm} q$ is well rounded, meaning that $||[\mathbf{w}]_{\mathtt{Lst}}||_\infty < 2^{\mathtt{Lst}-1} - E$ and $||\mathbf{w}||_\infty < \lfloor q/2 \rfloor - E$ should hold. When one of the checks fail, the signing procedure is restarted by sampling a new $\mathbf{y}$. When eventually both checks pass, the signature $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$ is output.

Verify (Algorithm 25). Signature verification is pretty lightweight and straightforward for this type of signature. Taking as input the message m, signature $\mathsf{sig} = (\mathbf{z}, \mathbf{c})$ and public key $pk = (seed_a, \mathbf{t})$, it works as follow: First, it generates the public parameter $\mathbf{a}$, then computes $\mathbf{w} = \mathbf{a} \cdot \mathbf{z} - \mathbf{t} \cdot \mathbf{c}$ and accepts the signature if $\mathbf{z} \in R_{q,[B-S]}$ and $\mathbf{c} \neq \mathsf{Enc}(\mathsf{hash}([\mathbf{w}]_{\mathtt{Mst}}, m))$.

3.5.2 Masking qTESLA signature scheme

In the process of masking qTESLA, we decided to make slight modifications in the signing procedure in order to facilitate masking. The idea is that some design elements providing small efficiency gains may be really hard to carry on to the masked version and actually do even more harm than good. Our two main modifications are the modulus which is chosen as the closest power of two of the original parameter set and the removal of the PRF to generate the polynomial $\mathbf{y}$.

Power of two modulo. Modular arithmetic is one of the core components of many cryptographic schemes. While, in general, it is reasonably fast for any modulus (but not necessarily straightforward to do in constant time), modular arithmetic in masked form is very inefficient, and it is often one of the bottlenecks in terms of running time. The gadget sec_{+_q} (See Gadget 7) is defined to add two integers in boolean masked form modulo q. The idea is to perform the addition over the integers naively and to subtract q if the value is larger than q. While this works completely fine, the computational overhead is large in practice and avoiding those reductions would drastically enhance execution time as noticed in [Mig+19] (See Section 3.3.6). The ideal case is to work over $\mathbb{Z}_{2^n}$. In this case, almost no reductions are needed throughout the execution of the algorithm and, when needed, can be simply performed by applying a mask on boolean shares. The reason why working with a power of two moduli is not the standard way to instantiate lattice-based cryptography is that it removes the possibility to use the number theoretic

transform (NTT) to perform efficient polynomial multiplication in $\mathcal{O}(n \log n)$. Instead, multiplication of polynomial has to be computed using the Karatsuba/Toom-Cook algorithm, which is slower for parameters used in state-of-the-art algorithms. Nevertheless, in our case, not having to use the heavy SecAddModp gadget largely overshadows the penalty of switching from NTT to Karatsuba. Since modulus for both parameter sets were already close to a power of two, we rounded to the closest one, i.e. 2^{22} for `qTESLA-I` and 2^{23} for `qTESLA-III`. This modification does not change the security of the scheme. Indeed, security-wise, for the heuristic version of the scheme that we study, we need a q such that $q > 4B$ [6] and the corresponding decisional LWE instance is still hard. However, the form of q does not impact the hardness of the problem as shown in [LS12a] and, since q was already extremely close to a power of two for both parameters sets, the practical bit hardness of the corresponding instance is not sensibly changed.

Removal of the PRF. It is well known that in Schnorr-like signatures, a devastating attack is possible if the adversary gets two different signatures using the same $\mathbf{y}$. Indeed, they can simply compute the secret $\mathbf{s} = \frac{\mathbf{z}-\mathbf{z}'}{\mathbf{c}-\mathbf{c}'}$. While such a situation is very unlikely due to the large size of $\mathbf{y}$, a technique to create a deterministic version of the signature was introduced in [MRa+99]. The idea is to compute $\mathbf{y}$ as $\mathsf{PRF}(secret\_seed, m)$ such that each message will have a different value for $\mathbf{y}$ unless a collision is found in PRF. This modification acts as a protection against very weak entropy sources but is not necessary to the security of the signature and was not present in ancestors of qTESLA. Unfortunately, adding this determinism also enabled some side-channel attacks [Pod+17; GP18]. Hence, the authors of qTESLA decided to take the middle ground by keeping the deterministic design but also seeding the oracle with a fresh random value r[7].
While those small safety measures certainly make sense if they do not incur a significant performance penalty, we decided to drop them and simply sample $\mathbf{y}$ at random at the beginning of the signing procedure. The reason is twofold. First, keeping deterministic generation of $\mathbf{y}$ implied masking the hash function evaluation itself, which is really inefficient if not needed and would unnecessarily complicate the masking scheme. Second, implementing a masking countermeasure is, in general, making the hypothesis that a reasonable source of randomness (or at least not weak to the point of having a nonce reuse on something as large as $\mathbf{y}$) is available to generate shares and thus can also be used for the signature itself.

Masked sign The masked signature can be found in Algorithm 27. After generating the public parameter $\mathbf{a}$ with the original GenA procedure, the gadget UGen (See Gadget 10) is used to get polynomials $\mathbf{y}_i$ such that $\mathbf{y} = \sum_{i=0}^{N} \mathbf{y}_i$ belongs to $R_{q,B}$. Then, thanks to the distributive property of the multiplication of ring elements, we can compute $\mathbf{v} = \mathbf{a} \cdot \mathbf{y} = \sum_{i=0}^{N} \mathbf{a} \cdot \mathbf{y}_i$ using regular polynomial multiplication, without relying on any complex gadget. The polynomial $\mathbf{c}$ is computed using the subroutine MaskedHash which is using the Rnd gadget (See Gadget 20) to compute qTESLA's rounding and hashing on a masked polynomial. Similarly to other Fiat–Shamir with aborts signatures (like GLP, Dilithium or qTesla), the computation of the hash function does not have to be performed in masked form since the knowledge of its inputs does not impact the security. Once $\mathbf{c}$ has been computed, the candidate signature can be computed directly on shares with the masked private key as $\mathbf{z} = \mathbf{y} + \mathbf{s} \cdot \mathbf{c} = \sum_{i=0}^{N} \mathbf{y}_i + \mathbf{s}_i \cdot \mathbf{c}$. Using the RS and WRnd gadgets (See Gadgets 18 and 22), the security and correctness parts of the signature follow trivially.

[6] The other condition on q in the parameters table of the submission is to enable the NTT
[7] Note that the fault attacks is still possible in case of failure of the RNG picking r

Once all checks have been passed, the signature can be safely unmasked using sec_{+_q} (See Gadget 7).

Algorithm 26 — Masked Hash

Data: The n coefficients $a^{(j)}$ to hash, in arithmetic masked form $(a_i^{(j)})_{0\leq i\leq d}$ and the message to sign m
Result: Hash of the polynomial c

Let u be a byte array of size n
for $j = 1$ *to* n **do**
  $u_j \leftarrow \mathsf{Rnd}((a_i^{(j)})_{0\leq i\leq d}, \mathtt{Tail})$
end
$c := \mathsf{hash}(u, m)$
$\mathbf{c} := \mathsf{Enc}(c)$
return c

Algorithm 27 — qTESLA masked signature algorithm

Data: message m, private key $sk = ((\mathbf{s}_i)_{0\leq i\leq d}, (\mathbf{e}_i)_{0\leq i\leq d})$, seed sd
Result: Signature $\mathsf{sig} = (\mathbf{z}_{unmasked}, \mathbf{c})$

$\mathbf{a} := \mathsf{GenA}(sd)$
$(\mathbf{y}_i)_{0\leq i\leq d} \leftarrow \mathsf{UGen}(B)$
for $i = 0, \ldots, d$ **do**
  $\mathbf{v}_i := \mathbf{a} \cdot \mathbf{y}_i$
end
$\mathbf{c} := \mathsf{MaskedHash}((\mathbf{v}_i)_{0\leq i\leq d}, m)$
$(\mathbf{z}_i)_{0\leq i\leq d} := \mathsf{MAdd}(\mathbf{c}, (\mathbf{s}_i)_{0\leq i\leq d}, (\mathbf{y}_i)_{0\leq i\leq d})$
if $rs := \mathsf{RS}((\mathbf{z}_i)_{0\leq i\leq d}, B - S) = 0$ **then**
  Goto 2
end
$(\mathbf{w}_i)_{0\leq i\leq d} := \mathsf{MAdd}(\mathbf{c}, (\mathbf{e}_i)_{0\leq i\leq d}, (\mathbf{v}_i)_{0\leq i\leq d})$
if $r := \mathsf{WRnd}\left((\mathbf{w}_i)_{0\leq i\leq d}, q/2 - E, 2^{\mathtt{Tail}-1} - E, \mathtt{Tail}\right) = 0$ **then**
  Goto 2
end
$\mathbf{z}_{unmasked} := \mathsf{Full} +_q ((\mathbf{z}_i)_{0\leq i\leq d})$
return $(\mathbf{z}_{unmasked}, \mathbf{c})$

Masked key generation The number of signature queries per private key can be high (up to 2^{64} as required by the NIST competition). In contrast, the key generation algorithm is typically only executed once per private key. The performance of the masked key generation is, therefore, less critical. We present here an (inefficient) masked version of the key generation algorithm that can be found in Algorithm 28. One can remark that the bottleneck gadget, GGen_σ (See Gadget 12), needs to make T comparisons for each coefficient of the polynomial which goes to a total of $T \cdot n$ comparisons for the whole generation. With a value of T around 200, sampling from the table is actually sensibly heavier than signing. Thus, our goal with this masked qTESLA key generation is to prove that masking without changing the design is costly but still doable. However, for a practical implementation in which the key generation might be vulnerable to side channels, one could prefer changing the design of the scheme. For example, Dilithium or GLP generate

the keys uniformly at random on a small interval and thus avoid this issue. One downside of these faster key generation is that the parameters of the scheme should be adapted in order to avoid having too many rejections in the signing algorithm.

In Algorithm 28, the element $\mathbf{a}$ is generated in unmasked form because it is also part of the public key. Then, $\mathbf{s}$ and $\mathbf{e}$ are drawn using the gadget GGen_σ. Another gadget MChk (See Gadget 23) is also introduced in the key generation. It checks that the sum of the h largest entries (in absolute value) is not above some bounds that can be found in Table 3.3. Then the public key $\mathbf{t}$ is computed in masked form and securely unmasked with the $\mathsf{Full}+_q$ gadget.

Algorithm 28 — Masked qTesla Key Generation

Result: $sk = ((\mathbf{s}_i)_{0\leq i\leq d}, (\mathbf{e}_i)_{0\leq i\leq d}, sd)$, $pk = (sd, \mathbf{t})$
pre-seed $\xleftarrow{r} \{0,1\}^\kappa$
sd $\coloneqq \mathsf{PRF}$(pre-seed)
$\mathbf{a} \coloneqq \mathsf{GenA}(sd)$
repeat
  $(\mathbf{s}_i)_{0\leq i\leq d} \leftarrow \mathsf{GGen}_\sigma()$
until $chk1 := \mathsf{MChk}((\mathbf{s}_i)_{0\leq i\leq d}, h, S) \neq 0$
repeat
  $(\mathbf{e}_i)_{0\leq i\leq d} \leftarrow \mathsf{GGen}_\sigma()$
until $chk2 := \mathsf{MChk}((\mathbf{e}_i)_{0\leq i\leq d}, h, E) \neq 0$
initialize $(sign_i^{\mathbf{s}})_{0\leq i\leq d}$ and $(sign_i^{\mathbf{e}})_{0\leq i\leq d}$ as two 1-bit arithmetic masking of either -1 or 1
$(\mathbf{s}_i)_{0\leq i\leq d} \leftarrow \mathsf{sec\&}\,((sign_i^{\mathbf{s}})_{0\leq i\leq d}, (\mathbf{s}_i)_{0\leq i\leq d})$
$(\mathbf{e}_i)_{0\leq i\leq d} \leftarrow \mathsf{sec\&}\,((sign_i^{\mathbf{e}})_{0\leq i\leq d}, (\mathbf{e}_i)_{0\leq i\leq d})$
$(\mathbf{t}_i)_{0\leq i\leq d} \coloneqq \mathsf{MAdd}(\mathbf{a}, (\mathbf{s}_i)_{0\leq i\leq d}, (\mathbf{e}_i)_{0\leq i\leq d})$
$\mathbf{t} \coloneqq \mathsf{Full} +_q ((\mathbf{t}_i)_{0\leq i\leq d})$
$sk \coloneqq ((\mathbf{s}_i)_{0\leq i\leq d}, (\mathbf{e}_i)_{0\leq i\leq d}, sd)$
$pk \coloneqq (sd, \mathbf{t})$
return sk, pk

3.5.3 EUF-CMA security in the d-probing model

We introduce a theorem that proves the d-`NIo` property of our masked key generation algorithm.

Theorem 10. *The key generation algorithm presented in Algorithm 28 is d-`NIo` secure with public outputs $chk1$ and $chk2$.*

Remark 14. *The knowledge of $chk1$ and $chk2$ cannot give more information on the outputted private key. Indeed, it only gives information on the number of attempts before outputting a key pair.*

Proof: The overall gadget decomposition of the key generation is in Fig. 3.10. We omit some of the non-sensitive values (like sd) for clarity. For simplicity and without losing generality, the theorem only considers one iteration of the key generation: the algorithm outputs $\perp$ if the checks are not validated. The public outputs $chk1$ and $chk2$ are then related to the number of iterations that is independent of the

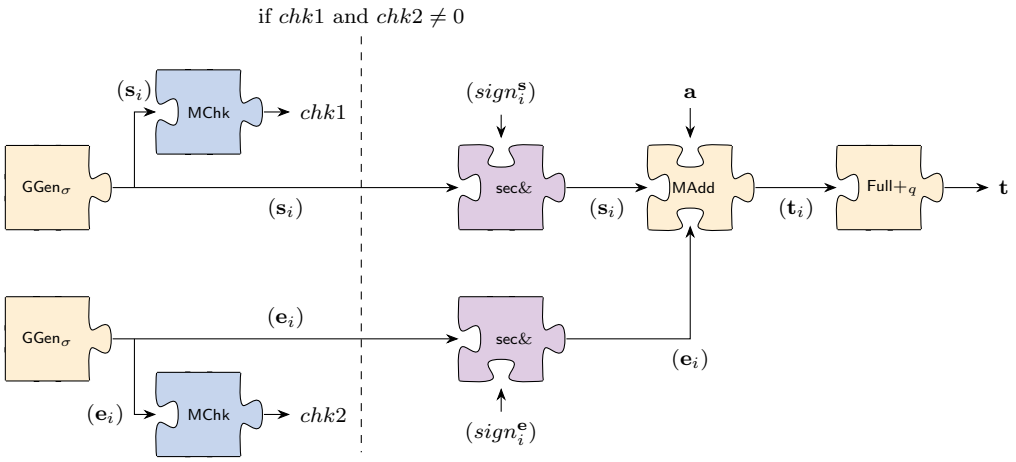


Figure 3.10: *Masked qTESLA Key Generation structure*

final secrets $\mathbf{e}$ or $\mathbf{s}$. The structure is very similar to the key generation of GLP in Algorithm 19, and there is no cycle in the structure. Thus, we can conclude that all the observations can be perfectly simulated with the only knowledge of the outputs.

□

In the following, we introduce a theorem that proves the d-NIo property of our masked signature algorithm. Here too, without losing generality, the theorem only considers one iteration for the signature: the signing algorithm outputs $\perp$ if one of the tests in Steps 9 or 13 in Algorithm 27 has failed. We denote by $\left(r^{(j)}\right)_{0\leq j<n}$,$\left(rs^{(j)}\right)_{0\leq j<n}$ and $\left(u^{(j)}\right)_{0\leq j<n}$ the outputs of RS, WRnd and Rnd (the values for each coefficient $j \in [0, n-1]$).

Theorem 11. *Each iteration of the masked signature in Algorithm 27 is d-NIo secure with public outputs*[8]

$$\left\{\left(r^{(j)}\right)_{0\leq j<n}, \left(rs^{(j)}\right)_{0\leq j<n}, \left(u^{(j)}\right)_{0\leq j<n}\right\}.$$

Proof: The overall gadget decomposition of the signature is in Fig. 3.11.

The gadget & multiplies each share of the polynomial $\mathbf{y}$ by the public value $\mathbf{a}$. By linearity, it is d-NI. Thus, all the sub-gadgets involved are either d-NI secure, d-NIo secure or they do not manipulate sensitive data (See Table 3.1). We prove that the final composition of all gadgets is d-NIo. As there is no cycle in the structure, the proof can be easily built from right to left similarly as for the proof of Theorem 6. □

Definition 14. *We denote by (r, rs, u)-qTESLA a variant of qTESLA where all the values*

$$\left\{\left(r^{(j)}\right)_{0\leq j<n}, \left(rs^{(j)}\right)_{0\leq j<n}, \left(u^{(j)}\right)_{0\leq j<n}\right\}$$

are outputted for each iteration during the signing algorithm.

[8]The number of iterations of the gadget UGen is omited as a public output.

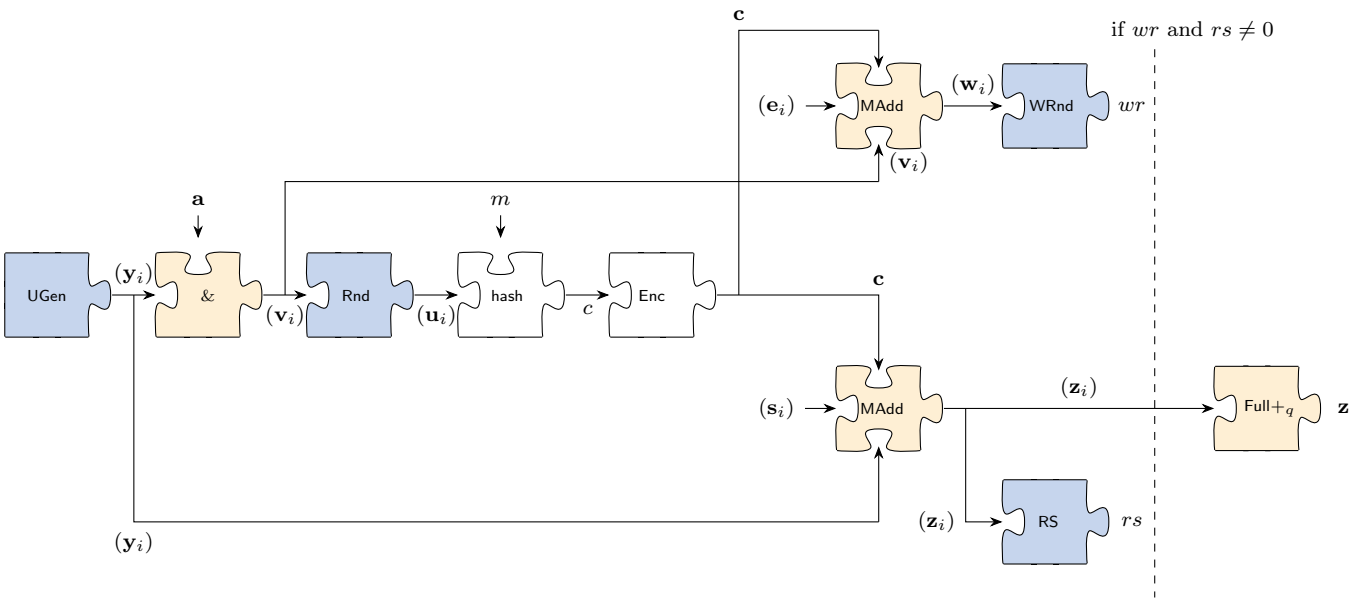


Figure 3.11: *Masked qTESLA Signature structure*

Theorem 11 allows reducing the EUF-CMA security in the d-probing model of our masked qTESLA signature at order d to the EUF-CMA security of (r, rs, u)-qTESLA. The security proof of qTESLA does not fully support (r, rs, u)-qTESLA because the adversary is not supposed to see the values of the signing's failed attempts. However, we can prove, with a Theorem similar to Theorem 5, that under some computational assumptions, outputting $\left(u^{(j)}\right)_{0\leq j<n}$ for each iteration does not affect the security. The values $\left\{\left(r^{(j)}\right)_{0\leq j<n}, \left(rs^{(j)}\right)_{0\leq j<n}\right\}$ correspond to rejection's conditions, and more precisely, the positions of the coefficients of the polynomials that do not pass the rejections. Such knowledge does not impact the scheme's security because the rejection probability does not depend on the position of the coefficients.

3.5.4 Optimized implementation and performance

This work was focused on *performance and reusability*, and François Gérard was the implementor. Our masked signature implementation still keeps the property of being compatible with the original verifying procedure of qTESLA and has been directly implemented within the code of the submission. Even if we target high order masking, we also implemented specialized gadgets for order 1 masking to provide a lightweight version of the masking scheme with a reasonable performance fitting nicely on embedded systems. We finally provided extensive performance data and showed that the cost of provable masking could be reasonable, at least for small orders.

We performed benchmarks for the two parameters sets `qTESLA-I` and `qTESLA-III`[9] on a laptop with a CPU Intel Core i7-6700HQ running at 2.60GHz as well as on a cortex-M4 microcontroller for the masking of order 1.

[9] As explained in Remark 13, the heuristic parameter sets on which our experiments are based, are now removed by the qTESLA team. We emphasize that the parameters we use for the implementation were *not* broken but are not part of the standardization process anymore. Furthermore, our theoretical work is somewhat oblivious to the underlying parameter set used to instantiate the signature, and the code can be adapted to implement the provably-secure sets as well.

Randomness We performed the tests with and without the random number generator activated (in gadgets). The reason why we decided to switch off the RNG[10] is to show how masking schemes of this magnitude are sensitive to the speed at which the device is capable of retrieving randomness. We also tested the smaller parameter set at order 1 on a Cortex-M4 microcontroller to see how it performs on a device more realistically vulnerable to side-channel attacks. We speculate that the scaling difference between the microcontroller and the computer is due to the fact that architectural differences matter less for the masking code than for the base signature code. Our tests with the randomness enabled were performed using xoshiro128** [BV18], a really fast PRNG that has been recently used to speed-up public parameters generation in a lattice-based cryptosystem [Bos+18b]. An implementor who is looking for our technique's real-life application and believes that masking needs a strong randomness would maybe want to use, instead, a cryptographically secure PRNG. Another option could be to expand a seed with the already available `cSHAKE` function, but as we will see in the sequel, it might be pretty expensive as the number of random bytes required grows very fast with the number of shares.

Individual gadgets. The result for individual gadgets over 1 000 000 executions can be found in Table 3.4. The table is divided into two parts: the top part contains measurements for the signing gadgets implementing functionalities of the signature and the bottom part contains measurements for the base gadgets implementing elementary operations. Unsurprisingly, we see that the most expensive signing gadget is WRnd. Indeed, it has to perform two absolute value computations in addition to two comparisons. Nevertheless, an actual substantial overall gain of performances would rather come from an improvement of the conversion from arithmetic to boolean masking since it is the slowest base gadget and is used in all signing gadgets. Furthermore, it should also be pointed out that most gadgets have a non-negligible dependency on the speed of sec& since it is called multiple times in sec_+, which itself appears multiple times in signing gadgets.

Signing procedure. The results for the full signature are given in Table 3.5. Since a large portion of the execution time is spent in calls to the random number generator, we decided to benchmark with and without the PRNG. The mention RNG off means that `rand_uint32()` was set to return 0. The mention RNG on means that `rand_uint32()` was set to return the next value of xoshiro128**. The purpose is to give an idea of how the algorithm itself is scaling, regardless of the speed at which the device is able to provide randomness. At the same time, the discrepancy between the values with and without the RNG underlines how masking schemes of this magnitude are sensitive to randomness sampling. In Table 3.7, we also computed the average number of calls to `rand_uint32()` to see how much randomness is needed for each order. Each call is retrieving a uniformly random 32-bit integer. As expected, this number is growing fast when the masking order is increased. The results for the masked signature at order 1 on cortex-M4 microcontroller are given in Table 3.6. We speculate that the scaling difference between the microcontroller and the computer is due to the fact that architectural differences matter less for the masking code than for the base signature code. Furthermore, we can see that `qTESLA-III` is scaling better than `qTESLA-I`. In addition to the natural variance of the experiments, we explain this result by the fact that increasing the masking order reduces the impact of the polynomial multiplication on the timing of the whole signature in favor of masking operations. Factoring out polynomial operations, `qTESLA-III` is scaling better because

[10] To switch the RNG off, we just set the `rand_uint32()` function to return 0

Table 3.4: *Median speed of principal gadgets in clock cycles over 1000000 executions*

Masking order	Order 1	Order 2	Order 3	Order 4	Order 5
UGen	98	410	840	1 328	2 416
Rnd	164	1 400	2 454	4 314	6 142
WRnd	280	2 080	3 914	6 432	9 034
RS	178	1 440	2 496	4 432	6 254
sec_{+_q}	44	294	592	870	1 192
sec&	20	28	44	70	96
$\mathsf{A}_{\hat{q}}\!\rightsquigarrow\!\mathsf{B}$	96	786	1 152	3 148	3 500
$\mathsf{B}\!\rightsquigarrow\!\mathsf{A}_q$	20	42	108	288	884

Table 3.5: *Median speed of masked signature in kilo clock cycles over 10000 executions*

Masking order	Un-masked	Order 1	Order 2	Order 3	Order 4	Order 5
`qTESLA-I` (RNG off)	646	2 394	7 000	9 220	16 578	24 375
`qTESLA-I` (RNG on)	671	2 504	13 879	24 583	39 967	59 551
`qTESLA-I` (RNG on) Scaling	1	**×4**	**×21**	**×37**	**×60**	**×89**
`qTESLA-III` (RNG off)	1 253	4 295	9 946	14 485	25 351	34 415
`qTESLA-III` (RNG on)	1 319	4 880	21 932	33 521	59 668	83 289
`qTESLA-III` (RNG on) Scaling	1	**×3**	**×17**	**×25**	**×45**	**×63**

the probability of rejection for these parameters sets is lower than for `qTESLA-I`. Hence, even if n is twice as large, less than twice the masking operations are performed overall.

As noted in [Mig+19], the power of two moduli allows getting a reasonable penalty factor for low masking orders. Without such a modification, the scheme would have been way slower. Besides, our implementation seems to outperform the masked implementation of Dilithium as given in [Mig+19]. The timing of our order 1 masking for `qTESLA-I` is around 1.3 ms, and our order 2 is around 7.1 ms. This result comes with no surprise because the unmasked version of qTESLA already outperformed Dilithium. However, we do not know if our optimizations on the gadgets could lead to a better performance for a masked Dilithium.

Table 3.6: *Median speed of masked signature in kilo clock cycles over 1000 executions for* `qTESLA-I` *on cortex-M4 microcontroller*

Masking order	Unmasked	Order 1
`qTESLA-I` CortexM4	11 304	23 520

Table 3.7: *Average number of calls to* `rand_uint32()` *(×1000)*

Masking order	Order 1	Order 2	Order 3	Order 4	Order 5
`qTESLA-I`	86	1 383	2 762	4 924	7 638
`qTESLA-III`	115	1 827	3 722	6 482	10 006

3.6 Perspectives

This chapter has presented the masking techniques for lattice-based Fiat–Shamir with aborts signatures. Several key new problems were encountered:

1. **Expensive Mask conversions.** Most steps of the signing lattice algorithm involve linear operations on polynomials that can be cheaply masked using $\bmod q$ arithmetic masking. However, for some operations like the generation of the randomness or the rejection sampling, this representation is less convenient. Therefore, we must carry out conversions from Boolean masking to arithmetic masking and the other way around.

2. **Public outputs.** For performance reasons, some intermediate variables may be revealed to the attacker. The scheme must then be proved secure with these public outputs. Besides, we introduced a new notion of security to account for public outputs.

3. **"Masking Friendliness".** Some sets of parameters are easier to mask than others. For example, by choosing a power of two as modulus, one can get better performance. Now that this is demonstrated providing one set of parameters that are more "masking-friendly" within the NIST competition could be an asset for the candidates.

Perspective 4. *A vital perspective would be to improve masked Gaussian Sampling (See Gadget 11). Indeed, even with recursion techniques, the cumulative distribution table seems heavy to carry in a masked form. This is what keeps us from masking the Falcon signature scheme, which is the only hash-and-sign signature from the competition.*

Part II

Cryptanalysis results

Chapter

4 Decryption failure attacks

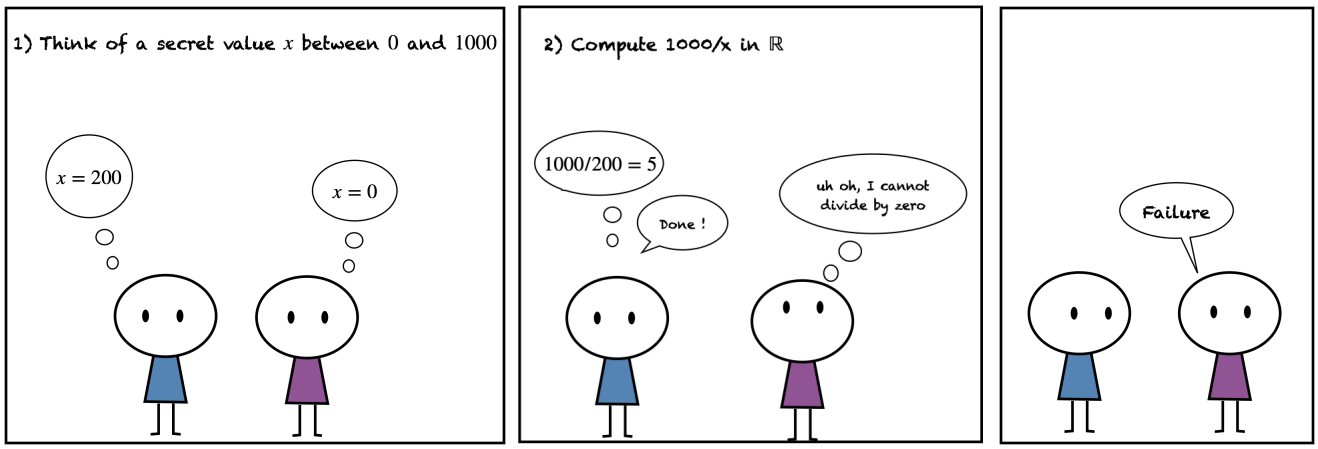


Figure 4.1: *Decryption failure attacks consist in analyzing the success or failure of an algorithm. Sometimes, an algorithm can generate an –extremely rare– problematic intermediate value that ends up with a failure. In practice, failures are very rare and if they happen, the algorithm is simply restarted. However, this knowledge can trigger attacks. For example, the purple algorithm chooses $x = 0$ which is the only value that cannot be inverted in $\mathbb{R}$. In that case, an attacker can conclude that the secret value was 0.*

The study of this chapter focuses on decryption failure attacks on lattice-based public key encryption or key exchanges. Guénaël Renault, one of my advisors at ANSSI suggested this topic to me at the beginning of my PhD. It was a perfect entry into the field of lattice-based KEMs. We started with the original lattice-based encryption scheme [LPR10], and we progressively understood the additional features that lead to NewHope [Alk+16b] then to the NIST lattice-based key exchange submissions that directly inherited the framework [Pöp+19; Nae+19; Sch+19]. The study was done through the prism of the *decryption failures attacks*. The latter family of attacks has been originally suggested by Fluhrer in [Flu16] against a reconciliation-based key exchange *à la* [Pei14]. With members of the ANSSI team: Aurélie Bauer, Henri Gilbert and Guénaël Renault, we found a way to adapt Fluhrer's attack to the recent decryption-based NewHope in a passively secure setting with key reuse [Bau+19]. Although our attack did not infringe any of the designed security claimes for NewHope, the main message of the paper was to warn the potential users on possible misuse contexts (see Section 4.1.3 for a definition)where the key is reused in scheme secure against passive attacks. It enables powerful decryption failures attacks.

Later on, a new family of generic attacks on schemes with stronger security models were introduced [DAn+19a]: they were no longer targeting the schemes in a misuse setting. After a post-quantum spring school in Oxford in 2019, with Jan-Pieter D'Anvers, Alessandro Bussoni, Henri Gilbert and Fernando Virdia, we started to consider a new hypothesis: the possible bootstrapping of the search for one failure in order to generate

other ones. The key idea was to consider the possible decryption failures in a geometric way. A few months later, Jan-Pieter D'Anvers, Fernando Virdia and I wrote the results of this study in a paper [DRV20] and we were very proud by the implications of the results. On the one hand, our paper show that after obtaining only one failure, anyone can find other ones "for free" and thus break the security. On the other hand, our paper greatly improves the state of the art concerning the multitarget attack model. Our student paper "(One) failure is not an option" has been accepted to Eurocrypt 2020[1].

Chapter content

This chapter highlights the implications of the non-perfect correctness feature that is inherent in lattice-based key exchanges. It seems that it can lead to vulnerabilities in various settings. Even though there is no total break of any of the NIST candidates, we do not know for sure that a nonzero failure probability can be harmless in any use case.

4.1 Introduction and motivations

4.1.1 Key Encapsulation Mechanisms (KEM)

As opposed to signature schemes, this chapter will study the security of lattice-based Key Encapsulation Mechanisms (or KEMs). Informally, with a KEM, one can generate an encapsulated key and send it to a public channel. The receiver, that knows a private key, is able to recover the key from its encapsulated version.

Definition 15. *A* Key Encapsulation Mechanism KEM *is a triple of algorithms containing*

1. *a probabilitic key generation algorithm, denoted* KeyGen*, that returns a public/private key pair* (pk, sk)*;*

[1] We refer to our animated Eurocrypt video https://www.youtube.com/watch?v=5zg2N82LpRs for a presentation of the geometric idea of the paper.

2. *a probabilitic encapsulation algorithm, denoted* Encaps*, that takes pk as input and generates an encapsulated key ct_{key} and a key m_{key};*
3. *a deterministic decapsulation algorithm, denoted* Decaps *that takes as input sk and ct_{key} and returns m_{key} or $\perp$, denoting failure.*

A KEM *is correct if for* $(sk, pk) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$,

$$P\left[\mathsf{KEM.Decaps}(sk, ct_{\text{key}}) \neq m_{\text{key}} : (ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{KEM.Encaps}(pk)\right]$$

is negligible in the security parameter. We say that it is perfectly correct *if the latter probability is zero for all* $(sk, pk) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$.

Key Encapsulation Mechanisms can be built from a better known type of mechanism, namely Public Key Encryption mechanisms (PKE). The difference between a PKE and a KEM is slight: in a nutshell, instead of outputting a key m_{key}, a chosen message m is given as input. Thus, the encapsulated key ct_{key} becomes a ciphertext ct. When considering KEMs, we will sometimes abusively use the term "message" for the common key m_{key} and "ciphertext" for the encapsulated key ct_{key}. The NIST call specifically asks for quantum-resistant KEM proposals.

4.1.2 Security properties

We can define several security properties for key encapsulation mechanisms that are inherited from the security properties of the public key encryptions.

Security Model 4 (IND-CPA security [KL14])**.** *A* KEM $=$ (KeyGen, Encaps, Decaps) *is IND-CPA, or "passively secure" if any probabilistic polynomial time adversary A has a negligible advantage of winning in the IND-CPA$_{A,\mathsf{KEM}}(1^\lambda)$ game presented in Security Game 4.*

Adversary		**Challenger**
	$\xleftarrow{\mathsf{KEM=(KeyGen,Encaps,Decaps)}}$	
	$\xleftarrow{pk}$	$(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ $(ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{Encaps(pk)}$ $b \xleftarrow{\$} \{0,1\}$ $m'_{\text{key}} = m_{\text{key}}$ if $b = 0$, $m'_{\text{key}} \xleftarrow{\$} \{0,1\}^n$ otherwise
	$\xleftarrow{pk, ct_{\text{key}}, m'_{\text{key}}}$	
	$\xrightarrow{b'}$	$b' = b$

Security Game 4: *IND-CPA game for KEM.*

IND-CPA is the basic security requirement for the KEM designers. We now introduce a stronger security property where a decapsulation oracle is provided. We assume that the maximum number of ciphertexts that can be queried to it for each key pair is

$$\mathtt{Q} = 2^K;$$

where in practice, $K = 64$ is often considered [NIS16, Section 4.A.2]. In this chapter, we keep the maximum number of queries as a parameter with no specific value in order to provide an adaptable granularity in the security assessment. Indeed, to mount an attack, the adversary trades off between the total number of queries and the work.

Security Model 5 (IND-CCA security [KL14]). *A* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is IND-CCA, or "actively secure", if any probabilistic polynomial time adversary A has a negligible probability of winning in the* $\text{IND-CCA}_{A,K}(1^\lambda)$ *game presented in Security Game 5.*

Adversary		Challenger
	$\xleftarrow{\mathsf{KEM}=(\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})}$	
		$(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
		$(ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
		$b \xleftarrow{\$} \{0,1\}$
		$m'_{\text{key}} = m_{\text{key}}$ if $b = 0$, $m'_{\text{key}} \xleftarrow{\$} \{0,1\}^n$ otherwise
	$\xleftarrow{pk, ct_{\text{key}}, m'_{\text{key}}}$	
2^K queries	$\xrightarrow{ct^{(1)}_{\text{key}} \neq ct_{\text{key}}}$	
		$u^{(1)} := \mathsf{Decaps}(sk, ct^{(1)}_{\text{key}})$
	$\xleftarrow{u^{(1)}}$	
	$\vdots$	
	$\xrightarrow{ct^{(2^K)}_{\text{key}} \neq ct_{\text{key}}}$	
		$u^{(2^K)} := \mathsf{Decaps}(sk, ct^{(2^K)}_{\text{key}})$
	$\xleftarrow{u^{(2^K)}}$	
distinguishing	$\xrightarrow{b'}$	$b' = b$

Security Game 5: *IND-CCA game for KEMs.*

Both properties originally existed for public-key encryption schemes (PKE). Lyubashevsky, Peikert and Regev [LPR10] introduced a simple protocol to build passively secure encryption from the Ring-LWE problem, inspired from the Diffie-Hellman key exchange [DH76] and ElGamal public-key encryption [ElG85]. Others were later introduced in [Pei14; Din12]. Naturally, the protocol can also be adapted to work based on LWE with no ring structure (Hard Problem 3) and on Mod-LWE (Hard Problem 5). In order to obtain active security, designers usually use an off-the-shelf CCA compiler, usually a (post-quantum) variant [Den03; HHK17; TU16; SXY17; Jia+17] of the Fujisaki-Okamoto transform [FO13] (FO). These come with proofs of security in the (quantum) random oracle model, with explicit bounds about the loss of security caused by the transformation.

4.1.3 Attack model

One commonly occurring characteristic of lattice-based KEM schemes is the lack of perfect correctness. In other words, for $(sk, pk) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$,

$$P\left[\mathsf{KEM.Decaps}(sk, ct_{\text{key}}) = m_{\text{key}} : (ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{KEM.Encaps}(pk)\right] < 1.$$

This means that sometimes, ciphertexts generated honestly using a valid public key may lead to decapsulation failures under the corresponding private key. Throughout this thesis, we'll refer to such events as "key mismatches", "failures", "decryption failures" or "decapsulation failures".

Misuse on IND-CPA schemes. While in practice, schemes are parametrized in such a way that decryption failures do not undermine overall performance, these can be leveraged as a vehicle for key recovery attacks against the key pair used to generate them. Jaulmes and Joux [JJ00] described such attacks against NTRU, after which it was extended in [How+03] and [GN07]. A similar attack on Ring-LWE based schemes was later presented by Fluhrer [Flu16] and extended by Băetu et al. [Băe+19]. It is also shown in [Ber+17a] that the private key of the round 1 NIST candidate HILA5 can be recovered following Fluhrer's approach. The aforementioned attacks all use specially crafted, dishonestly derived, ciphertexts that increase the probability of failure. Section 4.2 presents an attack that fits in this framework.

Leveraging the decryption failures is only possible if the scheme is not IND-CCA (see Security Model 4) and the private key is kept for several queries. More precisely, these attacks are in a *misuse* situation where the same key pair (sk, pk) is reused for multiple key establishments by the private key owner in an IND-CPA implementation. While slightly less powerful than a chosen ciphertext attack where a decryption oracle is available, attacks leveraging decryption failures still belong to the active attack category. Admittedly, it is not surprising that an active attack can be successful on a passively secure scheme because these attacks are outside of their security models. The study in Section 4.2 is motivated by the belief that an in-depth understanding of the security offered by candidate KEM mechanisms submitted to the NIST call for proposals in key reuse situations is a useful part of their cryptanalytic evaluation, even for those candidates for which key reuse is considered as a misuse of the mechanism in an IND-CPA implementation. Having an accurate estimate of the number of decryption queries and the complexity of the private key recovery really helps to assess the possible danger[2].

Generic attack techniques for IND-CCA schemes. The aforementioned attacks can be prevented with a transformation that achieves chosen ciphertext security (see Security Model 5). It stops the adversary from being able to freely maleate ciphertexts. Indeed, the randomness used during the encapsulation is generated by submitting the message m_{key} (and sometimes also the public key) to a random oracle. As this procedure is repeatable with knowledge of the message m_{key}, one can check the validity of ciphertexts ct_{key} during decapsulation. Hence, an adversary who wants to generate custom ephemeral ciphertexts ct_{key} would need to know a preimage of the appropriate random coins for the random oracle.

Therefore, their only option is to mount a Grover's search by randomly generating ciphertexts corresponding to different messages (see Section 1.1.2 for details on the quantum Grover's algorithm). The authors of the NIST Post-Quantum candidate Kyber [Sch+19] noted that it is possible to select ciphertexts with higher failure probability than average. D'Anvers et al. [DVV18] extended this idea to an attack called "failure boosting", where ciphertexts ct_{key} with higher failure probability are generated in a first offline phase. Submitting only the selected "weak" ciphertexts speeds the search for decryption failures. [DVV18] provides an analysis of the effectiveness of the attack on several NIST candidates. At the same time, Guo et al. [GJN19] described a similar generic adaptive attack that they applied against the IND-CCA secure ss-ntru-pke variant of NTRUEncrypt [Zha+19], which used an adaptive search for decryption failures exploiting infor-

[2] A similar need to investigate the resilience of candidate algorithms in misuse situations was encountered in the framework of the CAESAR competition aimed at selecting authenticated encryption primitives. In that competition, much analysis was conducted on the resistance of candidates to key recovery attacks in misuse cases such as nonce or decryption misuse (i.e. allowing nonce re-use in the leakage of the decryption failure oracles), and this provided quite useful information for the algorithms selection process.

mation from previously collected ciphertexts. The latter two papers were merged into [DAn+19a]. Section 4.3 presents an extension of the latter technique that bootstraps the search for failures based on [DRV20]. Basically, our paper shows that, under certain hypothesis, *even one* failed decryption per private key can result in significantly easier further decryption failures and thus key recovery attacks. Our paper also tackles the multitarget setting where an attacker can aim at many key pairs in order to break at least one of them.

This chapter unites two works [Bau+19] and [DRV20] that were chronologically separated. The first work, presented in Section 4.2, was initiated in 2017 while the second one, presented in Section 4.3, started in 2019 after the introduction of the stronger generic attacks in [DAn+19a]. In this thesis, we improve (and simplify) the results of [Bau+19] in Section 4.2 by combining them with the results of Chapter 5.

4.1.4 Key mismatch oracle

A decryption failure attack is a key recovery attack that leverages the lack of perfect correctness of the target scheme. In this section, we detail the access to the decryption failure information. The private key owner will be referred to as Alice. Note that Alice is also the party who initiates the two-round key establishment by sending her public key. Alice performs a key generation $(sk, pk) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$ and she keeps her key pair for many exchanges. Alice's private key sk is the target of the attack. A client, denoted Bob, can query Alice for making a key exchange using $(ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{KEM.Encaps}(pk)$ and sending ct_{key} to Alice. Alice thus derives the shared key with $m'_{\text{key}} := \mathsf{KEM.Decaps}(sk, ct_{\text{key}})$. Bob derives his symmetric keys based on the common shared key m_{key}. If Alice is later able to decrypt (and respond) based on m'_{key}, her version of m_{key}, it means that no decryption failure happened.

Moreover, in the opposite case, if $m'_{\text{key}} \neq m_{\text{key}}$; the shared symmetric keys are different between Alice and Bob, Alice will not be able to respond, and Bob will notice a decryption failure. We introduce an oracle to capture this bit of information (success or failure of the key exchange). We use the generic name of key mismatch oracle.

Definition 16 (key mismatch oracle)**.** *Let* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *with a message space* $\mathcal{M}$ *and a ciphertext space* $\mathcal{C}t$*. For all key pairs* $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, a key mismatch oracle is an oracle that outputs a bit corresponding to the success or failure of the decapsulation:*

$$\forall ct_{\text{key}} \in \mathcal{C}t, m_{\text{key}} \in \mathcal{M} \qquad \mathcal{O}_{sk}(ct_{\text{key}}, m_{\text{key}}) = \begin{cases} 1 \text{ if } \mathsf{Decaps}(sk, ct_{\text{key}}) = m_{\text{key}} \\ 0 \text{ otherwise.} \end{cases}$$

This definition can be identically translated to a PKE.

4.2 An example of decryption failure attack on an IND-CPA scheme

With its strong performance and its RLWE based security, NewHope is a high profile candidate of the NIST competition. Studying its security under several attacker models seems relevant.

Algorithm 29 — Key Encoding

Data: $\nu \in \{0,1\}^{256}$
$\mathbf{D} := 0$
for $i := 0$ *to* 255 **do**
  $\mathbf{D}_i := \nu_i.4s$
  $\mathbf{D}_{i+256} := \nu_i.4s$
  $\mathbf{D}_{i+512} := \nu_i.4s$
  $\mathbf{D}_{i+768} := \nu_i.4s$
end
return $\mathbf{D}$

Algorithm 30 — Key Decoding

Data: $\mathbf{D} \in R_q$
$\nu := 0$
for $i := 0$ *to* 255 **do**
  $t := \sum_{j=0}^{3} \left| \mathbf{D}_{i+256 \cdot j} - 4s \right|$
  if $t < q$ **then** $\nu_i := 1$ **else**
  $\nu_i := 0$
end
return ν

4.2.1 NewHope KEM

Parameters Let q be a prime number. Depending on the context, the elements in $\mathbb{Z}_q$ can be equivalently represented as integers in $\{0, \ldots, q-1\}$ or in $\{-\lfloor \frac{q-1}{2} \rfloor, \ldots, \lfloor \frac{q-1}{2} \rfloor\}$. The notation R_q refers to the polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$ with n a power of 2. If $\mathbf{P}$ belongs to R_q, it is a polynomial of degree at most $(n-1)$ with coefficients $\mathbf{P}^{(i)}$ belonging to the set $\mathbb{Z}_q$. Such elements can also be represented as vectors whose i-th coordinate is the coefficient related to x^i. Let us define ψ_κ the centered binomial distribution of parameter κ: one may sample from ψ_κ for integer $\kappa > 0$ by computing $\sum_{i=1}^{\kappa} b_i - b'_i$, where the b_i,$b'_i \in \{0,1\}$ are uniform independent bits. Its standard deviation is $\sqrt{\kappa/2}$.

Property 1. The elements generated according to a centered binomial distribution ψ_k of parameter κ are in the interval $[-\kappa, \kappa]$. In NewHope, $\kappa = 8$; thus, the coefficients of the small elements are in $[-8, 8]$.

By convention, for $a \in \mathbb{Z}$, we define $Sign(a)$ as *positive* (i.e "+1") when $a \geq 0$ and as *negative* (i.e "−1") when $a < 0$. If $x \in \mathbb{R}$, the integer $\lfloor x \rceil$ is defined as $\lfloor x + \frac{1}{2} \rfloor \in \mathbb{Z}$.

NewHope [Pöp+19] is a Ring-LWE based key establishment scheme (see Hard Problem 5) derived from NewHope-Usenix[Alk+16b], that is simplified because it does not use the reconciliation anymore. In this section, we describe NewHope, where we omit some details (e.g. the so-called NTT transform or the encoding of the messages) to simplify the presentation. This does not imply any loss of generality for our attack. To ease the understanding, we will describe the CPA-KEM version of NewHope in this section as the key mismatch oracle can be easily derived.

The polynomial ring R_q used in NewHope has the following parameters: $(n, q) = (1024, 12289)$ or $(n, q) = (512, 12289)$. The coefficients of the small elements drawn from R_q follow a centered binomial distribution ψ_κ^n with $\kappa = 8$. The standard deviation is $a = \sqrt{\frac{8}{2}} = 2$. We decided to focus on explaining the attack for $n = 1024$. Indeed, for $n = 512$ there is twice less redundancy, and the attack is easier. Thus, we fix $n = 1024$. These elements will be seen as vectors of size n with integer components. We denote $s = 1536$ which is such that $q = 8s + 1$. The aim of the system is to share a key of size $\frac{n}{4} = 256$ bits.

A public value $\mathbf{A} \in R_q$ is derived from a published *seed*. Four specific functions are introduced: Encode, Decode, Compress and Decompress. They are described in Algorithms 29 to 32. Note that we partly deviate from the notation of the original specification of these algorithms, since we use the parameter s (the original description is in [Alk+16a]). The following paragraphs describe these functions.

Algorithm 31 — Compression

Data: $\mathbf{C} \in R_q$
1 **for** $i := 0$ *to* $n-1$ **do**
2 $\quad \mathbf{c}[i] := \left\lceil \frac{8.\mathbf{C}[i]}{q} \right\rfloor \mod 8$
3 **end**
4 **return** $\mathbf{c}$

Algorithm 32 — Decompression

Data: $\mathbf{c} \in [0, 8[^n$
1 **for** $i := 0$ *to* $n-1$ **do**
2 $\quad \mathbf{C'}[i] := \left\lceil \frac{q.\mathbf{c}[i]}{8} \right\rfloor$
3 **end**
4 **return** $\mathbf{C'}$

Compress and Decompress.

The function Compress (Algorithm 31) takes as input a vector $\mathbf{C}$ in R_q and applies on each of its components a *modulus switching* to obtain an element $\mathbf{c}$ in $\mathbb{Z}_8[x]/(x^n+1)$. Compressing a vector $\mathbf{C}$ essentially means keeping the 3 most significant bits of each coefficient. The function Decompress (Algorithm 32) shifts the bits of the input $\mathbf{c} \in [0, 8[^n$ to place them among the most significant bits. These functions are not the inverse of each other.

Encode and Decode.

The Encode function takes a $n/4$-bit input ν with $n/4 = 256$ and creates an element $\mathbf{D} \in R_q$ which stores 4 times the element ν. The redundancy is used by the function Decode to recover ν with a noisy $\mathbf{D}$.

Let us now describe this scheme represented in Fig. 4.2.

1. *Setup*: Alice generates $\mathbf{S}$ and $\mathbf{E}$ with ψ_8. She sends her public key $pk := (seed, \mathbf{B} := \mathbf{AS} + \mathbf{E})$ to Bob.

2. *Key Encapsulation*: From a random *coin* acting as a seed[3], Bob derives 3 small secrets $\mathbf{S'}$, $\mathbf{E'}$ and $\mathbf{E''}$ in R_q and a random element ν_B of size $n/4$ which will be used to construct the encapsulated key. He computes $\mathbf{U} := \mathbf{AS'} + \mathbf{E'}$. He encodes ν_B into a redundant element $\mathbf{D}$ of R_q using the algorithm Encode (Algorithm 29). Bob uses Compress (Algorithm 31) to compress $\mathbf{C} := \mathbf{BS'} + \mathbf{E''} + \mathbf{D}$ into an element with very small coefficients as described above. He sends $ct_{\text{key}} := (\mathsf{Compress}(\mathbf{C}), \mathbf{U})$ to Alice. He deduces the shared secret as $m_{\text{key}} := \mathsf{hash}(\nu_B)$.

3. *Key Decapsulation*: Alice decompresses ct_{key} with Decompress into $\mathbf{C'}$(Algorithm 32). She computes $\mathbf{C'} - \mathbf{US}$ which is close to

$$\mathbf{C} - \mathbf{US} = \mathbf{ES'} + \mathbf{E''} + \mathbf{D} - \mathbf{E'S}. \tag{4.1}$$

Since $\mathbf{ES'} + \mathbf{E''} - \mathbf{E'S}$ is small, she recovers an estimated value ν_A of ν_B with a decoding algorithm called Decode(Algorithm 30). From ν_A, she can deduce $m'_{\text{key}} := \mathsf{hash}(\nu_A)$. Alice and Bob get the same key $m_{\text{key}} = m'_{\text{key}}$ with high probability.

Remark 15. *This Section presents NewHope-CPA-KEM, which is the target of this Section. Note that a PKE called NewHope-CPA-PKE has also been introduced in [Pöp+19].*

[3]This seed is not necessary in the IND-CPA version but it is used in the IND-CCA transformed version where Alice re-encapsulates for checking the validity of the ciphertext.

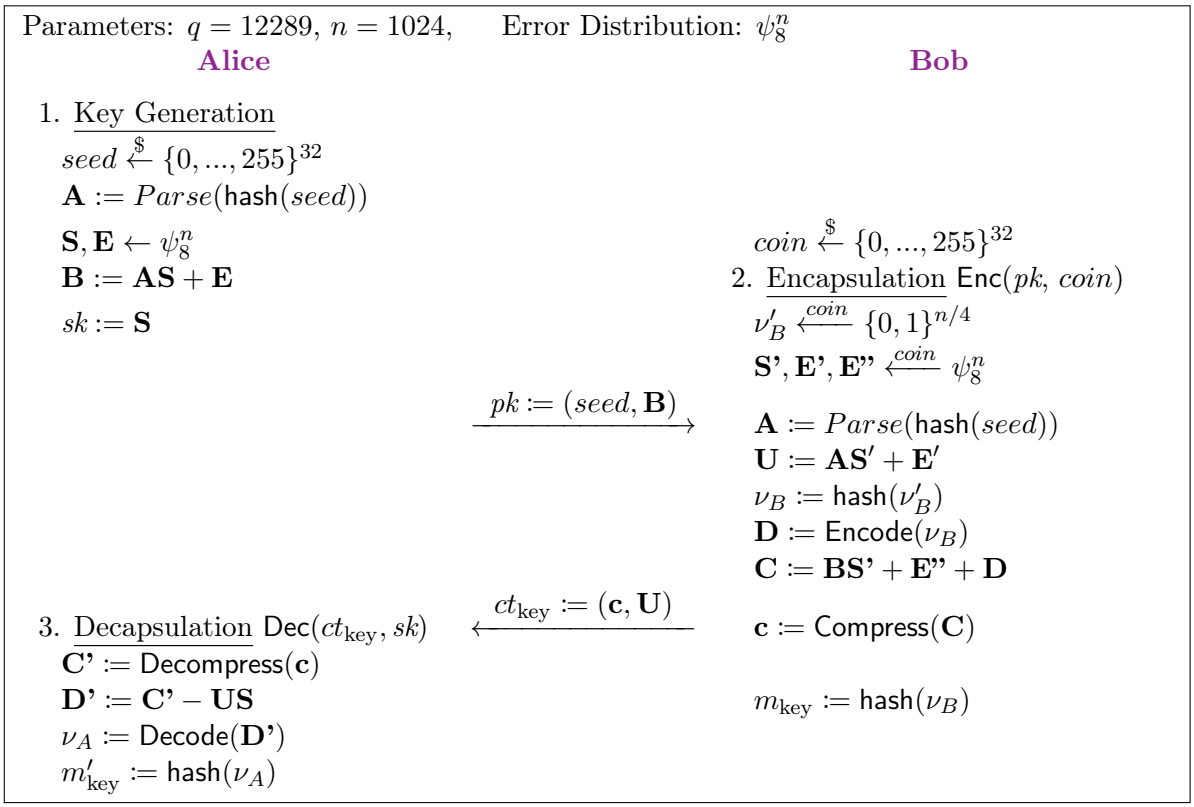


Figure 4.2: *Simplified NewHope*

4.2.2 Key mismatch oracle attack

We assume that NewHope is implemented in the misuse case described in Section 4.1.3. The use of the key mismatch oracle obviously leaks information on Alice's private key $\mathbf{S}$ as it is only passively secure. But the task of recovering $\mathbf{S}$ entirely seems much more complicated. Indeed as defined in Definition 16, the only information provided by the key mismatch oracle is a bit representing the success or mismatch of the key agreement. The difficulty for an attacker, called Eve, is to choose appropriate generated $(ct_{\text{key}}, m_{\text{key}})$ pairs dishonestly to get useful information on small parts of $\mathbf{S}$.

In a first step, Eve simplifies her part of the protocol in such a way that the knowledge of the key mismatch oracle output bit $\mathcal{O}_{sk}(ct_{\text{key}}, m_{\text{key}})$ can be easily exploited. To do so, she can fix, for instance, ν_E such that:

$$\nu_E := (1, 0, \ldots, 0) \text{ and thus } m_{\text{key}} := \mathsf{hash}(\nu_E). \tag{4.2}$$

The value of $\nu_E = (1, 0, \ldots, 0)$ has not been arbitrarily chosen; as we will see later, the 0 in positions 1 to 255 will help the success rate of the attack. From now on, the value of m_{key} is *fixed* according to Eq. (4.2). Moreover, if we replace ct_{key} by its definition: $ct_{\text{key}} = (\mathbf{c}, \mathbf{U})$, the oracle of Definition 16 can be reformulated using the oracle $\mathcal{O}^1$ defined below.

Definition 17. *Let us introduce oracle* $\mathcal{O}^1$ *such that* $\mathcal{O}^1_{sk}(\boldsymbol{c}, \boldsymbol{U}) := \mathcal{O}_{sk}\big((\boldsymbol{c}, \boldsymbol{U}), m_{\text{key}}\big)$

With this new definition, Eve can adapt the values of $\mathbf{c}$ and $\mathbf{U}$ to leverage Oracle $\mathcal{O}^1$ and retrieve information on $\mathbf{S}$. In other words, since m_{key} is fixed, the inputs $(\mathbf{c}, \mathbf{U})$ are the

degrees of liberty for finding $\mathbf{S}$.

From Alice's side, the link between ν_A and $\mathbf{S}$ passes through the functions Decode, Decompress (Algorithms 30 and 32) and the element $\mathbf{D'}$: $\nu_A = \mathsf{Decode}(\mathbf{D'}) = \mathsf{Decode}(\mathbf{C'} - \mathbf{US}) = \mathsf{Decode}(\mathsf{Decompress}(\mathbf{c}) - \mathbf{US})$. Thus, from the definition of the Decode algorithm, the value of $\nu_A[i]$, the i-th component of ν_A, is deduced from the following sign computation:

$$Sign\left(\sum_{j=0}^{3}\left|\left(\mathsf{Decompress}(\mathbf{c}) - \mathbf{U}\cdot\mathbf{S}\right)[i+256\cdot j] - 4s\right| - q\right) \tag{4.3}$$

We recall here that 0 is positive by convention.

The problem for Eve is that she is unable to know the number of errors that will occur at the end of the decryption computations and the positions in which they appear. Indeed, the key mismatch oracle only gives one bit of information corresponding either to *mismatch* or *success*. If there is a mismatch, Eve knows that *at least* one bit of ν_A is different from ν_E, but she can not determine which one (or which ones). Therefore, in order to mount an effective attack, Eve needs to restrict all these different possibilities by making the following hypothesis:

Assumption 2. For i from 1 to 255, the component $\nu_A[i]$ is equal to 0.

If Assumption 2 is verified, any failure in the communication comes from a single error in ν_A located in the very first component $\nu_A[0]$. Indeed, in that case, the success of the exchange only depends on the first computed value $\nu_A[0]$. In particular, if we assume this hypothesis, the oracle $\mathcal{O}_2$ depends only on the $\nu_A[0]$, and we obtain the following result.

Lemma 17. *Under Assumption 2, the initial oracle $\mathcal{O}^1$ can be rewritten as*

$$\mathcal{O}^1_{sk}(\boldsymbol{c}, \boldsymbol{U}) = \mathcal{O}^1_{\boldsymbol{S}}(\boldsymbol{c}, \boldsymbol{U}) = Sign\left(\sum_{j=0}^{j=3}\left|\left(\mathit{Decompress}(\boldsymbol{c}) - \boldsymbol{US}\right)[0+256\cdot j] - 4s\right| - q\right)$$

For mounting her attack, Eve has to find pairs $(\mathbf{c}, \mathbf{U})$ that

1. target the smallest number of bits of $\mathbf{S}$
2. verify Assumption 2

For item 1, the Decode algorithm takes coefficients of $\mathbf{S}$ four by four, and thus the size of the smallest target is a quadruplet of coefficients of $\mathbf{S}$. Actually, for a given quadruplet of integers $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3)$ and a target index k (i.e. an index corresponding to the components of $\mathbf{S}$ that Eve wants to retrieve), by taking

$$\mathbf{U} = \frac{s}{2}x^{-k} \quad \text{and } \mathbf{c} = \sum_{j=0}^{3}\Big((\ell_j + 4) \mod 8\Big)\cdot x^{256\cdot j} \tag{4.4}$$

one can prove (see in Proposition 5) that Eve targets the quadruplet $\big(\mathbf{S}[k+256\cdot j]\big)_{j=0\cdots 3}$. Indeed, the element x^{-k} will "rotate" $\mathbf{S}$ to target $\big(\mathbf{S}[k+256\cdot j]/2\big)_{j=0\cdots 3}$ and $\mathbf{c}$ is induced by the quadruplet $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3)$ that can vary.

About item 2, with the choice of Eq. (4.4), Assumption 2 is not validated if there exist $k \in [0, 255]$ such that

$$Sign\left(\sum_{j=0}^{j=3}\left|4+\mathbf{S}[k+256\cdot j]/2\right|-8\right)<0 \tag{4.5}$$

A probability analysis can be done by listing of all quadruplets in $[-k, k]$ that verify Eq. (4.5). Only a few unlikely quadruplet (e.g. [-8, -8, -8, -8]) verify it. For the considered set of parameters, Assumption 2 is correct for 98% of the private keys.

We can now introduce $\mathcal{O}^2$, a reformulation of $\mathcal{O}^1$ depending on target index k and the quadruplet $\underline{\ell}$ (see Eq. (4.4)):

$$\mathcal{O}^2_{\mathbf{S}}(k,\underline{\ell}) \coloneqq \mathcal{O}^1_{\mathbf{S}}\left(\frac{s}{2}x^{-k},\ \sum_{j=0}^{3}\Big((\ell_j+4)\mod 8\Big)\cdot x^{256\cdot j}\right)$$

Proposition 5. ***Final oracle.*** *Let us assume that Assumption 2 is verified. Let k be a target index ($k \in [0, 255]$). For a given integer quadruplet $\underline{\ell}$ in $[-4, 3]^4$, the $(\boldsymbol{c}, \boldsymbol{U})$ is explicit in Eq. (4.4) is such that*

$$\begin{aligned}\mathcal{O}^2_{\boldsymbol{S}}(k,\underline{\ell}) &= Sign\left(\textstyle\sum_{j=0}^{j=3}\left|\ell_j-\frac{\boldsymbol{S}[k+256\cdot j]}{2}+\frac{1+Sign(\ell_j)}{2s}\right|-8-\frac{1}{s}\right)\\ &\underset{s\gg 0}{\approx} Sign\left(\textstyle\sum_{j=0}^{j=3}\left|\ell_j-\frac{\boldsymbol{S}[k+256\cdot j]}{2}\right|-8\right)\end{aligned}$$

The quadruplet $\underline{\ell}$ is restricted to $[-4, 3]^4$ so that $\ell_j + 4 \in [0, 7]$. Indeed, by definition, the Decompress function (Algorithm 32) takes its inputs in $\mathbb{Z}_8$. In the next section, we explain how to effectively use the form $\mathcal{O}_2$ of the key mismatch oracle to extract the secret $\mathbf{S}$.

Proof: Let k be an index defined in $[0, 255]$ and let us fix $\underline{\ell} = (\ell_0, \ell_1, \ell_2, \ell_3) \in [-4, 3]^4$. We show how the explicit values for $(\mathbf{c}, \mathbf{U})$ given in Equation 4.4 provide the expected result.

Let us compute the values $\mathsf{Decompress}(\mathbf{c})[256 \cdot j]$:

$$\begin{aligned}\mathsf{Decompress}(\mathbf{c})[256\cdot j] = \left\lceil\frac{(\ell_j+4 \mod 8)\times q}{8}\right\rfloor &= \left\lceil(\ell_j+4 \mod 8)s+\frac{(\ell_j+4 \mod 8)}{8}\right\rfloor\\ &= (\ell_j+4 \mod 8)s+\left\lceil\frac{(\ell_j+4 \mod 8)}{8}\right\rfloor\\ &= \begin{cases}(\ell_j+4 \mod 8)s \text{ if } \ell_j<0\\ (\ell_j+4 \mod 8)s+1 \text{ if } \ell_j\geq 0\end{cases}\end{aligned}$$

Under Assumption 2 the equation given by Lemma 17 thus becomes:

$$Sign\left(\sum_{j=0}^{j=3}\left|(\ell_j+4 \mod 8).s-s\frac{\mathbf{S}[k+nj]}{2}-4s+\frac{1+Sign(\ell_j)}{2}\right|-(8s+1)\right)$$

Since $\underline{\ell}$ belong in $[-4, 3]^4$, we can remove the modulo and divide by s to obtain the final oracle.

□

4.2.3 Key recovery

With Proposition 5, one can see that the secret $\mathbf{S}$ is directly in jeopardy. In fact, the output of the oracle greatly depends on the coefficients of the secret, and it is possible to manipulate them separately by groups of four. Several iteration methods on $\underline{\ell}$ can be performed to recover the coefficients of $\mathbf{S}$, and the real challenge is to minimize the number of queries to the key mismatch oracle.

In [Bau+19], we present a technique that consists in studying the variation of a function $f_{\ell_1,\ell_2,\ell_3}(\ell_0) = \mathcal{O}^2_{\mathbf{S}}(k,(\ell_0,\ell_1,\ell_2,\ell_3))$ for fixed k,ℓ_1,ℓ_2,ℓ_3 (where the three last integers are fixed randomly and $k \in [0,255]$). The minimum of this function corresponds to $\ell_0 = \mathbf{S}[k]/2$. In our paper [Bau+19], we considered $\lfloor\cdot\rfloor$ by mistake instead of $\lceil\cdot\rfloor$ in the Decompress function. Consequently, the term $\frac{1+Sign(\ell_j)}{2s}$ was not taken into account. It led to an underestimation of the number of queries necessary to recover the full secret, originally estimated at around $18,500$ queries. Indeed, the term $\frac{1+Sign(\ell_j)}{2s}$ actually leads to some wrong candidates for finding the minimum of the function f_{ℓ_1,ℓ_2,ℓ_3} which complicates the attack. In [QCD19], the authors claimed that the attack actually needs more redundancy in the search for the minimum and thus that more queries are needed for a full key recovery. In this thesis, in view of the results of Chapter 5, we take a different path that does not increase the number of queries previously estimated. The technique to query the oracle remains exactly the same, but the obtained data is analyzed through the prism of Chapter 5. More precisely, in Section 5.6.2, we will show that there exists an attack 80 bits below the claimed security with the same number of queries as estimated in [Bau+19], namely $18,500$ average queries.

Experiment This work is still supported by the proof of concept coded in Magma CAS in 2017 [BCP97][4]. We ran 1000 experiments and we were able to run the attack on 95% of the private keys (because of Assumption 2). The latter attack outputs a candidate for the private key within $17,000$ queries on average. This candidate is close to the secret and can be used for decreasing lattice-reduction attacks as shown in Section 5.6.2.

4.3 (One) failure is not an option: an improved attack on a CCA scheme

Let us now introduce [DRV20]. Contrary to the decryption failure attack on the passively secure NewHope KEM presented in Section 4.2, a generic attack cannot use specially crafted ciphertexts (as in Eq. (4.4)).

The target: a generic scheme.

[4]The Magma code can be found at https://www.di.ens.fr/~mrossi/

Algorithm 33 — PKE.KeyGen

Result: (sk, pk)
$\mathbf{A} \xleftarrow{\$} R_q^{l \times l}$
$\mathbf{s}, \mathbf{e} \in R_q^l \leftarrow D^l_{\mathbb{Z},\sigma_s} \times D^l_{\mathbb{Z},\sigma_e}$
$\mathbf{b} := \mathbf{s}\mathbf{A}^T + \mathbf{e}$
return $(sk := \mathbf{s}, pk := (\mathbf{b}, \mathbf{A}))$

Algorithm 34 — PKE.Enc

Data: $pk = (\mathbf{b}, \mathbf{A}), \mathbf{m} \in \mathcal{M}; coin$
Result: a ciphertext ct
$\mathbf{s}', \mathbf{e}' \in R_q^l \xleftarrow{coin} D^l_{\mathbb{Z},\sigma_s} \times D^l_{\mathbb{Z},\sigma_e}$
$\mathbf{e}'' \in R_q \xleftarrow{coin} D_{\mathbb{Z},\sigma_e}$
$\mathbf{b}' := \mathbf{s}'\mathbf{A} + \mathbf{e}'$
$\mathbf{v}' := \mathbf{b}\mathbf{s}'^T + \mathbf{e}'' + \lfloor q/2 \rfloor \cdot \mathbf{m}$
return $ct := (\mathbf{v}', \mathbf{b}')$

Algorithm 35 — PKE.Dec

Data: $sk = \mathbf{s}, ct = (\mathbf{v}', \mathbf{b}')$
Result: A message $\mathbf{m}'$
$\mathbf{m}' := \lfloor \lfloor 2/q \rfloor (\mathbf{v}' - \mathbf{b}'\mathbf{s}^T) \rceil$
return $\mathbf{m}'$

	l	n	q	σ_s	σ_e	Failure probability	Classical	Quantum
Chosen parameters	3	256	8192	2.00	2.00	2^{-119}	2^{195}	2^{177}
Saber	3	256	8192	1.41	2.29	2^{-136}	2^{203}	2^{185}
Kyber 768	3	256	3329	1.00	1.00/1.38†	2^{-164}	2^{181}	2^{164}

† Standard deviation of the error term in the public key and ciphertext respectively

Table 4.1: *Comparison between our target scheme and Saber and Kyber 768, as parametrized in Round 2 of the NIST PQC standardization process. The classical (resp. quantum) security is evaluated using the Core-SVP [Alk+16b] methodology, assuming the cost of BKZ with block size β to be $2^{0.292\beta}$ (resp. $2^{0.265\beta}$).*

For the need of the study, we had to consider a generic scheme based on Mod-LWE (see Hard Problem 5) *with no rounding and no error-correcting code*. Although no such implementation exists without these features, we take inspiration from two NIST candidates Saber [DAn+19b] and Crystals-Kyber [Sch+19] to design a simpler scheme with similar specifications. The parameters are chosen to provide similar resistance to known quantum and classical attacks while being simpler to analyse due to the lack of rounding and error-correcting code.

Let $q, n, l, \sigma_s, \sigma_z$ be public parameters, and we use the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ as defined in Eq. (1.3). We describe a general passively secure (Security Model 4) public key encryption scheme derived from [LPR10] based on Mod-LWE (Hard Problem 5) in Algorithms 33 to 35. In the latter, a random $coin \in \{0,1\}^{256}$ acts as a seed for drawing from Gaussian distributions. These Gaussian distributions follow Section 2.2.2 notations. The message space is defined as $\mathcal{M} :=$ {polynomials in R_q with coefficients in $\{0,1\}$}.

We show the corresponding IND-CCA secure (see Security Model 5) KEM Decapsulation and Encapsulation in Algorithms 36 and 37 using a CCA transformation derived from the Fujisaki-Okamoto transform [FO13]. hash_0 and hash_1 are hash functions. In the KEM, the ciphertext space is $\mathcal{C}t = (R_q^l, R_q)$ and the space of the common key m_{key}, called as "message space" is $\mathcal{M} = \{0,1\}^{256}$.

We selected a set of parameters to implement Algorithms 33 to 37. They ensure a

Algorithm 36 — KEM.Encaps

Data: pk
Result: a ciphertext ct_{key}, a common key m_{key}

$m \xleftarrow{\$} \{0,1\}^{256}$
$(\overline{m_{\text{key}}}; coin) := \mathsf{hash}_0(pk, m)$
$ct_{\text{key}} := \mathtt{PKE.Enc}(pk, m, coin)$
$m_{\text{key}} := \mathsf{hash}_1(\overline{m_{\text{key}}}, coin)$
return $(ct_{\text{key}}, m_{\text{key}})$

Algorithm 37 — KEM.Decaps

Data: sk, ct_{key}
Result: a common key m_{key}

$m' := \mathtt{PKE.Dec}(sk, ct_{\text{key}})$
$(\overline{m_{\text{key}}}', coin') := \mathsf{hash}_0(pk, m')$
$ct'_{\text{key}} := \mathtt{PKE.Enc}(pk, m'; coin')$
if $ct_{\text{key}} = ct'_{\text{key}}$ **then**
  return $m_{\text{key}} := \mathsf{hash}_1(\overline{m_{\text{key}}}', coin')$
else
  return $m_{\text{key}} := \perp$ `// Could return a pseudo-random string to implicitly reject`

similar failure probability and security to Kyber and Saber. These parameters can be found in Table 4.1. The security estimates are generated using the Core-SVP methodology [Alk+16b] and the LWE estimator[5] [Alb+18], while the failure probability of Kyber and Saber is given as reported in their respective NIST round 2 documentations [Sch+19; DAn+19b]. The failure probability of our chosen parameters is determined by calculating the variance of the error term and assuming the distribution to be Gaussian.

Remark 16. *We do not consider the case of "plain" LWE based schemes like FrodoKEM [Nae+19] or Round5 [Baa+19]. Nonetheless, we believe that the attack methodology could translate to the LWE setting as the failure condition and the failure probabilities are similar to the investigated case.*

Remark 17. *Several lattice-based candidates submitted to the NIST post-quantum cryptography standardization process use variants of the protocol by Lyubashevsky et al. [LPR10]. Deviating from the original design, most candidates perform an additional rounding of the ciphertext* $\mathbf{v}'$*, in order to reduce bandwidth. The designers of New Hope [Pöp+19] and LAC [Lu+19] choose to work directly over rings (or equivalently, they choose a module of rank* $l = 1$*) and add error correction on the encapsulated message, while the designers of Kyber [Sch+19] and Saber [DAn+19b] choose a module of rank* $l > 1$ *and perform an additional rounding of* $\mathbf{b}'$ *(and* $\mathbf{b}$ *in case of Saber). We here focus on the basic version given in Algorithms 33 to 37 and leave the study of the effect of compression for further work.*

Introduction of the failure boosting. The "failure boosting" is a technique introduced in [DAn+19a] to increase the failure rate of (Ring/Mod)-LWE/LWR based schemes by honestly generating ciphertexts and only querying weak ones, i.e. those that have a failure probability above a certain threshold $f_t > 0$. This technique is especially useful in combination with Grover's algorithm, in which case the search for weak ciphertexts can be sped up quadratically (see Section 1.1.2). The attack assumes that a key pair has been drawn by the target $(sk, pk) \leftarrow \mathsf{KEM.Keygen}(1^\lambda)$ and the attacker knows pk, and she wants to find sk. The failure boosting consists of two phases: a precomputation phase, and a phase where the decryption oracle is queried.

Precomputation phase. The adversary does an offline search for weak ciphertexts. For finding one weak ciphertext, the adversary has to proceed as follows.

[5] The estimator can be found at `https://bitbucket.org/malb/lwe-estimator/`.

1. Generate an encapsulation pair with $(ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{KEM.Encaps}(pk)$.
2. If $\mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) > f_t$, keep ct_{key} in a weak ciphertext list, otherwise go to Step 1.

In Step 2, $\mathcal{P}^{\text{fail}}_{pk}(.)$ is defined as the failure probability given a public key pk, a certain ciphertext ct_{key} and common shared key m_{key}. It is computed as follows.

$$\mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) \coloneqq \sum_{sk} P\left[\mathcal{O}_{sk}(ct_{\text{key}}, m_{\text{key}}) = 0\right] \cdot P[(sk, pk) \leftarrow \mathsf{KEM.Keygen}(1^\lambda)]. \quad (4.6)$$

Let D^{enc}_{pk} be the probability distribution of the output of $\mathsf{KEM.Encaps}(pk)$. In other words, for $ct_{\text{key}} \in \mathcal{C}t$ and $m_{\text{key}} \in \mathcal{M}$, $D^{\text{enc}}_{pk}(ct_{\text{key}}, m_{\text{key}}) \coloneqq P[(ct_{\text{key}}, m_{\text{key}}) \leftarrow \mathsf{KEM.Encaps}(pk)]$. The probability of finding a weak ciphertext can be expressed as follows:

$$\alpha_{f_t} \coloneqq \sum_{\left\{ \begin{array}{l} (ct_{\text{key}}, m_{\text{key}}) \in (\mathcal{C}t, \mathcal{M}) : \\ \mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) > f_t \end{array} \right\}} D^{\text{enc}}_{pk}(ct_{\text{key}}, m_{\text{key}}). \quad (4.7)$$

An adversary thus needs to perform on average $\alpha^{-1}_{f_t}$ work to obtain one weak ciphertext, or $\sqrt{\alpha^{-1}_{f_t}}$ assuming Grover's search achieves a full speed-up.

Decryption oracle query phase. After the precomputation phase, an adversary has a probability β_{f_t} that a weak ciphertext results in a failure, where β_{f_t} can be calculated as a weighted average of the failure probabilities of weak ciphertexts:

$$\beta_{f_t} \coloneqq \frac{\sum_{(ct_{\text{key}}, m_{\text{key}}) : \mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) > f_t} D^{\text{enc}}_{pk}(ct_{\text{key}}, m_{\text{key}}) \cdot \mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}})}{\sum_{(ct_{\text{key}}, m_{\text{key}}) : \mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) > f_t} D^{\text{enc}}_{pk}(ct_{\text{key}}, m_{\text{key}})}. \quad (4.8)$$

Thus to obtain one decryption failure with probability $1 - e^{-1}$, an adversary needs to perform approximately $\beta^{-1}_{f_t}$ queries and therefore $\alpha^{-1}_{f_t}\beta^{-1}_{f_t}$ work (or $\sqrt{\alpha^{-1}_{f_t}}\beta^{-1}_{f_t}$ using a quantum computer).

Application to the generic lattice-based scheme. Throughout this chapter, we use the extended definition for the norms as presented in Section 1.2.4. Following the execution of the protocol in Algorithms 36 and 37, for a private key[6] composed of $(\mathbf{s}, \mathbf{e})$ and a ciphertext ct_{key} and a common key m_{key} both generated from small elements $\mathbf{s}'$, $\mathbf{e}'$ and $\mathbf{e}''$, we get by construction

$$\mathcal{O}_{(\mathbf{s},\mathbf{e})}(ct_{\text{key}}, m_{\text{key}}) = 0 \iff \|\mathbf{e}\mathbf{s}'^T - \mathbf{s}\mathbf{e}'^T + \mathbf{e}''\|_\infty > q/4 \quad (4.9)$$

where we recall that $\mathcal{O}$ is defined in Definition 16. Thus,

$$\mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) = \sum_{(\mathbf{s},\mathbf{e})} P\left[\|\mathbf{e}\mathbf{s}'^T - \mathbf{s}\mathbf{e}'^T + \mathbf{e}''\|_\infty > q/4\right] \cdot P[(\mathbf{s}, \mathbf{e}, pk) \leftarrow \mathsf{KEM.Keygen}(1^\lambda)]. \quad (4.10)$$

The analysis of [DAn+19a] provides several assessments of α_{f_t} and β_{f_t} with several schemes. The better an adversary can predict $\mathcal{P}^{\text{fail}}_{pk}(ct, K)$, the more efficient failure boosting will be. With no information about the secret, except its distribution, an adversary is

[6] Here, for the ease of the writing, we consider that $\mathbf{e}$ is also part of the private key.

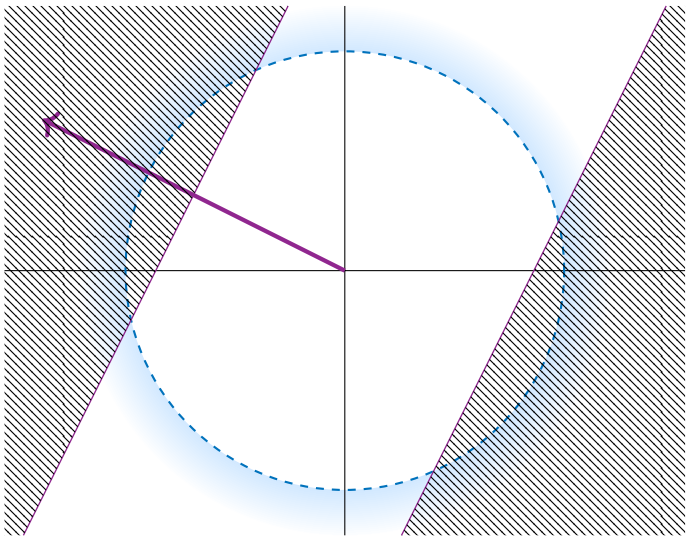


Figure 4.3: *Simplified diagram to represent the ciphertexts (in blue). The weak ones are defined as the ciphertexts with a probability higher than f_t.*

bound to standard failure boosting as presented above. For a graphical intuition, a two-dimensional toy example is depicted in Fig. 4.3, where the red arrow abusively represents the secret as a vector. Ciphertexts that lie in the dashed area will provoke a failure as the inner product with the secret vector will exceed the threshold q_t. The blue circle is a circle of ciphertexts that have a certain failure probability f_t as estimated by an adversary who does not know the secret. During the failure boosting procedure, we will generate random ciphertexts and only select the ciphertexts with a higher failure probability than f_t, i.e. that are outside the blue circle. One can graphically see in Fig. 4.3 that these ciphertexts will have a higher failure probability and a higher norm. Note that Fig. 4.3 is an oversimplified 2-dimension example that does not take into account the polynomial structure and the high dimensionality of the space.

4.3.1 Definitions on failure events

In this section, we rewrite the access to the bit of information given by the key mismatch oracle in an inequality in the scalar product. This expression in Eq. (4.9) can be simplified by defining the vector $\mathbf{S}$ as the vertical concatenation of $-\mathbf{s}$ and $\mathbf{e}$, the vector $\mathbf{C}$ as the concatenation of $\mathbf{e}'$ and $\mathbf{s}'$, and by replacing $\mathbf{e}''$ with $\mathbf{G}$, as shown below:

$$\mathbf{S} := \begin{bmatrix} -\mathbf{s} \\ \mathbf{e} \end{bmatrix} \quad \mathbf{C} := \begin{bmatrix} \mathbf{e}' \\ \mathbf{s}' \end{bmatrix} \quad \mathbf{G} := \mathbf{e}''. \tag{4.11}$$

For a more visual explanation, until the end of this chapter, $\mathbf{S}$ and $\mathbf{C}$ are written as column vectors. Here, $\mathbf{S}$ contains the secret elements of the private key and $\mathbf{C}$ and $\mathbf{G}$ consist of elements used to construct the ciphertext and common key[7]. By using these vectors, the error expression can be rewritten: a failure occurs when $\|\mathbf{S}^T\mathbf{C} + \mathbf{G}\|_\infty > q/4$.

The standard deviation of the terms in the polynomial $\mathbf{S}^T\mathbf{C}$ equals $\sqrt{2n}\sigma_s\sigma_e$, versus a standard deviation of σ_e for the terms of $\mathbf{G}$. Therefore, the influence of $\mathbf{G}$ on the failure rate is limited, i.e. $\|\mathbf{S}^T\mathbf{C} + \mathbf{G}\|_\infty \approx \|\mathbf{S}^T\mathbf{C}\|_\infty$. Let $q_t := q/4$ denote the failure threshold,

[7]When talking about ciphertexts, we will sometimes refer to their underlying elements $\mathbf{C}$ and $\mathbf{G}$.

we will consider that

$$\mathcal{O}_{\mathbf{S}}(ct_{\text{key}}, m_{\text{key}}) = 0 \iff \|\mathbf{S}^T\mathbf{C}\|_\infty > q_t. \tag{4.12}$$

However, with some extra work, one could have rewritten a more accurate Eq. (4.12) as $\|\mathbf{S}^T\mathbf{C}\|_\infty > q_t - \|\mathbf{G}\|_\infty$, and instead of considering q_t to be fixed, taken the distribution of $q_t - \|\mathbf{G}\|_\infty$ as shown in [DAn+19a]. For the ease of the implementation and due to the low influence of $\mathbf{G}$ on the failure rate, we prefer to stick with Eq. (4.12). We now introduce a more handy way of writing the failure condition (Eq. (4.12)) by only using vectors in $\mathbb{Z}_q$.

Definition 18 (Coefficient vector)**.** *For $\mathbf{S} \in R_q^{l\times 1}$, we denote by $\overline{\mathbf{S}} \in \mathbb{Z}_q^{ln\times 1}$, the representation of $\mathbf{S}$ where each polynomial is decomposed as a list of its coefficients in*[8] *$\mathbb{Z}_q$ starting by the low degree coefficients.*

Definition 19 (Rotations)**.** *For $r \in \mathbb{Z}$ and $\mathbf{C} \in R_q^{l\times 1}$, we denote by $\mathbf{C}^{(r)} \in R_q^{l\times 1}$, the following vector of polynomials*

$$\mathbf{C}^{(r)} \coloneqq X^r \cdot \mathbf{C}(X^{-1}) \mod X^n + 1.$$

Correspondingly, $\overline{\mathbf{C}^{(r)}} \in \mathbb{Z}_q^{ln\times 1}$ denotes its coefficient vector.

It is easy to show that $\overline{\mathbf{C}^{(r)}}$ is constructed as to ensure that for $r \in [0, ..., n-1]$, the r^{th} coordinate of $\mathbf{S}^T \cdot \mathbf{C}$ is given by the scalar product $\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}}$. In other words, one is now able to decompose $\mathbf{S}^T\mathbf{C}$ as a sum of products:

$$\mathbf{S}^T \cdot \mathbf{C} = \sum_{r\in[0,n-1]} \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} \cdot X^r. \tag{4.13}$$

One can observe that this construction is only valid for the modulo $X^n + 1$ ring structure, but it could be adapted for other ring structures. Note that for any $r \in \mathbb{Z}$, $\mathbf{C}^{(r+n)} = -\mathbf{C}^{(r)}$ and $\mathbf{C}^{(r+2n)} = \mathbf{C}^{(r)}$. Besides, taking into account the extension of the norms to vectors of polynomials, one can make the following remark.

Remark 18. *Note that for any $r \in \mathbb{Z}$, $\|\overline{\mathbf{C}^{(r)}}\|_2 = \|\overline{\mathbf{C}}\|_2 = \|\mathbf{C}\|_2$ and $\|\overline{\mathbf{C}^{(r)}}\|_\infty = \|\overline{\mathbf{C}}\|_\infty = \|\mathbf{C}\|_\infty$.*

The decomposition in Eq. (4.13) will allow a geometric interpretation of the failures as it will be shown in the rest of the chapter. First, let us introduce a brief example to illustrate Definitions 18 and 19.

Example 4.3.1. For a secret $\mathbf{S}$ and a ciphertext $\mathbf{C}$ in $\mathbb{Z}_q^{2\times 1}[X]/(X^3+1)$:

$$\mathbf{S} = \begin{bmatrix} s_{0,0} + s_{0,1}X + s_{0,2}X^2 \\ s_{1,0} + s_{1,1}X + s_{1,2}X^2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} c_{0,0} + c_{0,1}X + c_{0,2}X^2 \\ c_{1,0} + c_{1,1}X + c_{1,2}X^2 \end{bmatrix} \tag{4.14}$$

we get the following vectors:

$$\overline{\mathbf{S}} = \begin{bmatrix} s_{0,0} \\ s_{0,1} \\ s_{0,2} \\ s_{1,0} \\ s_{1,1} \\ s_{1,2} \end{bmatrix}, \quad \overline{\mathbf{C}^{(0)}} = \begin{bmatrix} c_{0,0} \\ -c_{0,2} \\ -c_{0,1} \\ c_{1,0} \\ -c_{1,2} \\ -c_{1,1} \end{bmatrix} \quad \overline{\mathbf{C}^{(1)}} = \begin{bmatrix} c_{0,1} \\ c_{0,0} \\ -c_{0,2} \\ c_{1,1} \\ c_{1,0} \\ -c_{1,2} \end{bmatrix} \quad \overline{\mathbf{C}^{(2)}} = \begin{bmatrix} c_{0,2} \\ c_{0,1} \\ c_{0,0} \\ c_{1,2} \\ c_{1,1} \\ c_{1,0} \end{bmatrix} \quad \overline{\mathbf{C}^{(3)}} = \begin{bmatrix} -c_{0,0} \\ c_{0,2} \\ c_{0,1} \\ -c_{1,0} \\ c_{1,2} \\ c_{1,1} \end{bmatrix} \dots$$

[8] Here, all the elements in $\mathbb{Z}_q$ are represented as integers belonging in $[-q/2, q/2]$.

Now, let us verify Eq. (4.13) by considering the coefficient of X^1 in $\mathbf{S}^T\mathbf{C}$:

$$s_{0,0}c_{0,1} + s_{0,1}c_{0,0} - s_{0,2}c_{0,2} + s_{1,0}c_{1,1} + s_{1,1}c_{1,0} - s_{1,2}c_{1,2} = \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(1)}}. \tag{4.15}$$

In case of a failure event, $\mathbf{S}^T\mathbf{C}$ satisfies Eq. (4.12). Therefore, at least one element among all the coefficients

$$\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(0)}}, \ldots, \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(2n-1)}}$$

is larger than q_t. Indeed, note that $\forall r \in [0, 2n-1] \quad \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} < -q_t \implies \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r+n)}} > q_t$.

Definition 20 (Failure event)**.** *A failure event will be denoted with F. More precisely, for $r \in [0, 2n-1]$, we denote by F_r the failure event where*

$$\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} > q_t.$$

The event F_r gives a two-fold information: it provides the location of the failure in the $\mathbf{S}^T\mathbf{C}$ polynomial, and it also provides *the sign* of the coefficient that caused the failure. Thus,

$$\mathcal{O}_{\mathbf{S}}(ct_{\text{key}}, m_{\text{key}}) = 1 \iff \exists r \in [0, 2n-1] \quad \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} > q_t \quad (F_r). \tag{4.16}$$

An assumption on the failing ciphertexts. In order to predict the results of our attack, we will make the following orthogonality assumption.

Assumption 3. Let $N \ll 2nl$, and $\mathbf{C}_0, \cdots, \mathbf{C}_N$ be ciphertexts that lead to failure events $F_{r_0}, \cdots, F_{r_N}$. The vectors $\overline{\mathbf{C}_0^{(r_0)}}, \cdots, \overline{\mathbf{C}_N^{(r_N)}}$ are considered orthogonal when projected on the hyperplane orthogonal to $\overline{\mathbf{S}}$.

This assumption is an approximation that is supported by the fact that normally-distributed vectors in high-dimensional space have a strong tendency towards orthogonality [CFJ13].

4.3.2 Directional failure boosting

Once $N \geq 1$ decryption failures $\mathbf{C}_0, \ldots, \mathbf{C}_{N-1}$ are found, additional information about the private key $\mathbf{S}$ becomes available, and can be used to refine the failure estimation for new ciphertexts and thus speed up failure boosting. We now introduce an iterative two-step method to perform directional failure boosting.

Step 1 An estimate, denoted $\overline{\mathbf{E}}$, of the 'direction' of the secret $\overline{\mathbf{S}}$ in $\mathbb{Z}_q^{ln}$ is obtained from $\mathbf{C}_0, \ldots, \mathbf{C}_{N-1}$.

Step 2 The estimate $\overline{\mathbf{E}}$ is used to inform the search for weak ciphertexts and improve the failure probability prediction for a new ciphertext $\mathbf{C}_N$. One is able to refine Eq. (4.9)'s criterion $\mathcal{P}_{pk}^{\text{fail}}(ct_{\text{key}}, m_{\text{key}}) \geq f_t$.

Once new failing ciphertexts are found in step 2, one can go back to step 1 and improve the estimate $\overline{\mathbf{E}}$ and thus bootstrap the search for new failures.

To give an intuition, a two-dimensional toy representation can be found in Fig. 4.4. Like in the classical failure boosting technique (presented before in Fig. 4.3), the red arrow depicts the secret $\overline{\mathbf{S}}$, while the additional blue arrow marks estimate $\overline{\mathbf{E}}$. Using

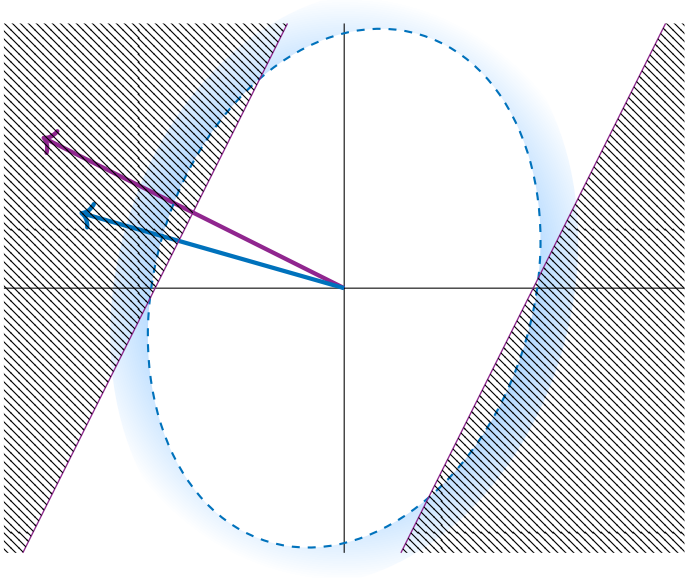


Figure 4.4: *The directional information allows a refined acceptance criterion, here represented as an ellipse.*

this estimate, we can refine the acceptance criterion to the depicted ellipse to reflect our knowledge about the secret better. Ciphertexts outside this ellipse will be flagged as weak ciphertexts, and while the probability of finding such a ciphertext is the same, the failure probability of weak ciphertexts is now higher. As in, more of the blue zone lies in the dashed area.

Step 1: Estimating the direction $\overline{\mathbf{E}}$

Informally, $\overline{\mathbf{E}}$ should be a vector that has approximately the same direction as $\overline{\mathbf{S}}$. Denoting the angle between $\overline{\mathbf{E}}$ and $\overline{\mathbf{S}}$ as θ_{ES}, the bigger $|\cos(\theta_{ES})|$, the closer our estimate is to $\pm\overline{\mathbf{S}}$ and the better our estimate of failure probability will be. Since we focus on estimating the direction of $\overline{\mathbf{S}}$, $\overline{\mathbf{E}}$ will always be normalized.

In this section, we derive an estimate $\overline{\mathbf{E}}$ of the direction of the secret $\overline{\mathbf{S}}$ given $N \geq 1$ ciphertexts $\mathbf{C}_0, \ldots, \mathbf{C}_{N-1}$. Our goal is to find $\overline{\mathbf{E}}$ such that $|\cos(\theta_{ES})|$ is as big as possible. We will first discuss the case where the adversary has one ciphertext, then the case where she has two, followed by the more general case where she has N ciphertexts.

One ciphertext. Assume that a unique failing ciphertext $\mathbf{C}$ is given. For a failure event F_r, $\overline{\mathbf{E}} = \overline{\mathbf{C}^{(r)}} / \left\|\overline{\mathbf{C}^{(r)}}\right\|_2$ is a reasonable choice as $\cos(\theta_{ES})$ is bigger than average. This can be seen as follows:

$$|\cos(\theta_{ES})| = \frac{\left|\overline{\mathbf{S}}^T \cdot \overline{\mathbf{E}}\right|}{\left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{E}}\right\|_2} = \frac{\left|\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}}\right|}{\left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2} > \frac{q_t}{\left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2}. \tag{4.17}$$

Keeping in mind that the cosine of angles between random vectors strongly tends to zero in high dimensional space, so that even a relatively small value of $|\cos(\theta_{ES})|$ might be advantageous.

Table 4.2: *Experimental probability of finding the correct relative rotations and thus building the correct estimate* $\overline{\mathbf{E}}$ *with the knowledge of* 2, 3, 4 *and* 5 *failing ciphertexts. This experiment has been performed by generating 1000 times each number of ciphertexts and trying to find the correct values of the* r_i.

	2 ciphertexts	3 ciphertexts	4 ciphertexts	5 ciphertexts
$P[r_i = \delta_{i,0}\ \forall i \in [1, N-1]]$	84.0%	95.6%	$> 99.0\%$	$> 99.0\%$

One can argue that it is not possible to compute $\overline{\mathbf{C}^{(r)}}$ without knowledge of r; but in the general case, the failure location is unknown. However, $\overline{\mathbf{E}} = \overline{\mathbf{C}^{(0)}} / \left\| \overline{\mathbf{C}^{(0)}} \right\|_2$ is an equally good estimate regardless of the value of r. Indeed, $\overline{\mathbf{C}^{(0)}}$ approximates a rotation of the secret $\overline{\mathbf{S}'} := \overline{X^{-r} \cdot \mathbf{S}}$ instead of $\overline{\mathbf{S}}$, which can be seen using the equality $\overline{\mathbf{A}}^T \cdot \overline{\mathbf{B}} = \overline{X^i \mathbf{A}}^T \cdot \overline{X^i \mathbf{B}}$:

$$\begin{aligned} \overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} &= \overline{X^{-r} \cdot \mathbf{S}}^T \cdot \overline{X^{-r} X^r \mathbf{C}^{(0)}} \\ &= \overline{X^{-r} \cdot \mathbf{S}}^T \cdot \overline{\mathbf{C}^{(0)}}. \end{aligned} \tag{4.18}$$

Furthermore, multiplicating a polynomial in R_q with a power of X does not change its infinity norm, as the multiplication only results in the rotation or negation of coefficients. Thus, using an estimate of the direction of $\overline{X^{-r} \cdot \mathbf{S}}$ is as good as an estimate of the direction of $\overline{\mathbf{S}}$ when predicting the failure probability of ciphertexts, and we can use $\overline{\mathbf{E}} = \overline{\mathbf{C}^{(0)}} / \left\| \overline{\mathbf{C}^{(0)}} \right\|_2$.

Multiple ciphertexts. Now, we assume that N linearly independent failing ciphertexts $\mathbf{C}_0, \ldots, \mathbf{C}_{N-1}$, resulting from failure events $F_{r_0}, \ldots, F_{r_{N-1}}$ respectively, are given. In [DRV20], we introduce a generalized method to recover the relative positions denoted $\delta_{1,0}, \ldots, \delta_{N-1,0}$ with high probablity. The technique is based on the "loopy belief propagation" algorithm [Pea14]. Basically, we represent the ciphertexts $\mathbf{C}_0, \ldots, \mathbf{C}_{N-1}$ as nodes in a fully connected graph. For two ciphertexts $\mathbf{C}_i$ and $\mathbf{C}_j$, for any $r \in [0, 2n-1]$, the inner product

$$\overline{\mathbf{C}_i^{(r+r_i)}}^T \cdot \overline{\mathbf{C}_j^{(r+r_j)}}$$

is slightly correlated to the secret because both $\mathbf{C}_i^{(r+r_i)}$ and $\mathbf{C}_j^{(r+r_j)}$ fail for the same rotated secret $\overline{X^{-r}\mathbf{S}}$. The algorithm takes random pairs of ciphertexts and propagate the best relative positions. It finally converges with outputting a $\delta_{1,0}, \ldots, \delta_{N-1,0}$. We refer to our paper for more details on the method. In Table 4.2, we show the experimental results obtained using this technique by Jan-Pieter D'Anvers, who is to be credited for the idea of loopy belief propagation.

Once these relative positions are found, they can be combined in an estimate $\overline{\mathbf{E}}$ with $\overline{\mathbf{E}} := \overline{\mathbf{C}_{\text{tot}}} / \left\| \overline{\mathbf{C}_{\text{tot}}} \right\|_2$ where

$$\overline{\mathbf{C}_{\text{tot}}} := \left(\overline{\mathbf{C}_0^{(0)}} / \left\| \overline{\mathbf{C}_0^{(0)}} \right\|_2 + \sum_{i \in [1, N-1]} \overline{\mathbf{C}_i^{(\delta_{i,0})}} / \left\| \overline{\mathbf{C}_i^{(\delta_{i,0})}} \right\|_2 \right) / N. \tag{4.19}$$

Step 2: Finding weak ciphertexts

In this section, we are given an estimate $\overline{\mathbf{E}}$ and we refine the acceptance criterion. Instead of accepting if $\mathcal{P}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}) \geq f_t$, our condition is slightly changed to a new condition denoted $\mathcal{Q}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}, \overline{\mathbf{E}}) > f_t$.

First, for $r \in [0, 2n-1]$, we will estimate the probability that a ciphertext leads to an error in the r^{th} location. Decomposing the vectors $\overline{\mathbf{S}}$ and $\overline{\mathbf{C}}$ in a component orthogonal to $\overline{\mathbf{E}}$, denoted with subscript $\perp$, and a component parallel to $\overline{\mathbf{E}}$, denoted with subscript $\|$, we obtain a failure expression. We write "$\mid \overline{\mathbf{E}}$" to represent the knowledge of the vector $\overline{\mathbf{E}}$.

$$
\begin{aligned}
P\left[\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} > q_t \mid \overline{\mathbf{E}}\right] &= P\left[\overline{\mathbf{S}_{\|}}^T \cdot \overline{\mathbf{C}^{(r)}_{\|}} + \overline{\mathbf{S}_{\perp}}^T \cdot \overline{\mathbf{C}^{(r)}_{\perp}} > q_t \mid \overline{\mathbf{E}}\right] \\
&= P\left[\begin{pmatrix} \left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2 \cos(\theta_{SE}) \cos(\theta_{C^{(r)}E}) + \\ \left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2 \sin(\theta_{SE}) \sin(\theta_{C^{(r)}E}) \cos(\theta_{SC^r}) \end{pmatrix} > q_t \mid \overline{\mathbf{E}}\right] \\
&= P\left[\cos(\theta_{SC^r}) > \frac{q_t - \left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2 \cos(\theta_{SE}) \cos(\theta_{C^rE})}{\left\|\overline{\mathbf{S}}\right\|_2 \left\|\overline{\mathbf{C}^{(r)}}\right\|_2 \sin(\theta_{SE}) \sin(\theta_{C^rE})} \mid \overline{\mathbf{E}}\right]
\end{aligned}
$$

where θ_{SE} and $\theta_{C^{(r)}E}$ are the angles of $\overline{\mathbf{S}}$ and $\overline{\mathbf{C}^{(r)}}$ with the estimate $\overline{\mathbf{E}}$ respectively, and where θ_{SC^r} is the angle between $\overline{\mathbf{S}}_{\perp}$ and $\overline{\mathbf{C}^{(r)}_{\perp}}$. We assume no other knowledge about the direction of the secret apart from the directional estimate $\overline{\mathbf{E}}$. In this case, using Assumption 3, θ_{SC^r} can be estimated as a uniform angle in a $(2nl-1)$-dimensional space. Then θ_{SC^r} is assumed to follow the random angle probability distribution (defined as Θ_{2nl-1} in [DRV20]).

The values $\overline{\mathbf{E}}$, $\|\mathbf{C}\|_2$ and $\cos(\theta_{C^{(r)}E})$ are known, meanwhile the values $\|\mathbf{S}\|$ and θ_{SE} can be modelled using their probability distribution. Thus, we can approximate the probability $P\left[\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} > q_t \mid \overline{\mathbf{E}}\right]$.

Assumption 4. We assume that failures at different locations are independent.

According to [DAn+19a], Assumption 4 is a valid assumption for schemes without error-correcting codes.

We can then calculate the failure probability of a certain ciphertext as:

$$
\mathcal{Q}^{\text{fail}}_{pk}(ct_{\text{key}}, m_{\text{key}}, \overline{\mathbf{E}}) \coloneqq 1 - \prod_{r=0}^{2n} \left(1 - P\left[\overline{\mathbf{S}}^T \cdot \overline{\mathbf{C}^{(r)}} > q_t \mid \overline{\mathbf{E}}\right]\right) \tag{4.20}
$$

As this expression gives us a better prediction of the failure probability of ciphertexts by using the information embedded in $\overline{\mathbf{E}}$, we can more accurately (Grover) search for weak ciphertexts and thus reduce the work to find the next decryption failure. Moreover, the better $\overline{\mathbf{E}}$ approximates the direction of $\overline{\mathbf{S}}$, the easier it becomes to find a new decryption failure.

Verification of the accuracy of the estimate

To verify the efficiency of the directional failure boosting, one must quantify the accuracy of the estimate $\overline{\mathbf{E}}$ computed according to Section 4.3.2. In this purpose, the cosine $\cos(\theta_{SE})$

Table 4.3: *Accuracy of the estimate derived from several failures. Expected value of* $\cos(\theta_{SE})$*. The closer to* 1*, the more accurate* $\overline{\mathbf{E}}$ *is.*

#failures	1	2	3	5	10	20	30	50	100
theoretical	0.328	0.441	0.516	0.613	0.739	0.841	0.885	0.926	0.961
experimental	0.318	0.429	0.502	0.600	0.727	0.832	0.878	0.921	0.958

(actually, it is $\cos(\theta_{S'E})$ for S' being a fixed rotation of the secret that can be recovered with $2N$ tests) can be experimentally computed. We refer to our paper for more information on the experiment [DRV20].

4.3.3 Finalizing the attack with lattice reduction

Once multiple failures are found, the private key can be recovered with lattice reduction techniques as presented in [DVV18, Section 4] and in [GJN19, Step 3 of the attack]. The following section simply outlines how their technique transposes to our framework. As shown in Section 4.3.2, an estimate $\overline{\mathbf{E}}$ of the direction of a rotated version $\overline{\mathbf{S}'} = \overline{X^r\mathbf{S}}$ with an unknown value r is provided. Therefore, similarly to [GJN19], an attacker can obtain an estimation of $\overline{\mathbf{S}'}$ (and not only its direction) by rescaling

$$\overline{\mathbf{E}'} := \overline{\mathbf{E}} \cdot Nq_t \cdot \left(\left\| \overline{\mathbf{C}_0^{(0)}} + \sum_{i\in[1,n-1]} \overline{\mathbf{C}_i^{(r_i)}} \right\|_2 \right)^{-1},$$

using the approximation $\overline{\mathbf{E}'}^T \cdot 1/N \left(\overline{\mathbf{C}_0^{(0)}} + \sum_{i\in[1,n-1]} \overline{\mathbf{C}_i^{(r_i)}} \right) \approx q_t$.

Then, for each possible $r \in [0, 2n-1]$, an attacker can perform lattice reduction and recover candidates for $\mathbf{s}, \mathbf{e}$ that are accepted if they verify $\mathbf{b} = \mathbf{As} + \mathbf{e}$. One caveat is that an attacker may have to run a lattice reduction up to $2n$ times. Since $\overline{\mathbf{E}'} - \overline{\mathbf{S}'}$ is small, the attacker can construct an appropriate lattice basis encoding $\overline{\mathbf{E}'} - \overline{\mathbf{S}'}$ as a unique shortest target vector and solves the corresponding Unique-SVP problem with the BKZ algorithm [SE94; CN11; Alk+16b; Alb+17]. The block size of BKZ will depend on the accuracy of the estimate $\overline{\mathbf{E}}$. Indeed, the standard deviation of $\overline{\mathbf{E}'}_i - \overline{\mathbf{S}'}_i$ is of the order of $\sigma_s \cdot \sin(\theta_{S'E})$ (assuming that $\theta_{S'E}$ is small and $\|\overline{\mathbf{S}'}\|_2 \approx \|\overline{\mathbf{E}'}\|_2$).

Anticipating Chapter 5, the information given by our estimate can be modeled by approximate hints : for all $i \in [0, ln-1]$,

$$\overline{\mathbf{S}'}_i + e = \overline{\mathbf{E}'}_i$$

where e follows a centered normal distribution of standard deviation $\sigma_s^2 \cdot \sin(\theta_{S'E})^2$.

Thus, when many decryption failures are available, $\sin(\theta_{S'E})$ gets very small, and the complexity of total key recovery including the primal lattice reduction is dominated by the work required for constructing $\overline{\mathbf{E}}$. For example, in the case of our toy scheme, if $\cos(\theta_{S'E}) > 0.985$, using the tool introduced in Chapter 5 (or [APS15]), the BKZ block size becomes lower than 363 which leads to less than 2^{100} quantum work (in the Core-SVP [Alk+16b] 0.265β model). As we will see in Section 4.3.4, this is less than the work required to find the first failure.

Remark 19. *One can think that the failures obtained by directional failure boosting will not be totally independent. Indeed, the failing ciphertexts are roughly following the same direction. However, applying our Assumption 3, in high dimensions, for a reasonable number n of failures ($N \ll 2ln$), the hypercone in which the failures belong is large enough in order for linear dependencies to happen with very low probability.*

4.3.4 Estimation of the attack performance

The key takeaway of this section is that, for Table 4.1 parameters, the more failing ciphertexts have been found, the easier it becomes to obtain the next one, and that most of the effort is concentrated in finding the first failure. The final work and query overheads are stored in Table 4.4.

In this section, we will derive the optimal work and queries for an adversary, in order to obtain N ciphertexts with probability $1 - e^{-1}$. We introduce the following notation: to find the $(i+1)^{\text{th}}$ failing ciphertext; the adversary performs Q_i queries. Using a Poisson distribution, the success probability of finding the $(i+1)^{\text{th}}$ ciphertext in Q_i queries is $1-e^{-Q_i\beta_{i,f_i}}$. The probability of obtaining N failures can then be calculated as the product of the success probabilities of finding ciphertexts 0 to $N-1$:

$$P_N = \prod_{i=0}^{N-1}(1 - e^{-Q_i\beta_{i,f_i}}). \tag{4.21}$$

This is a slight underestimation of the success probability of the attack because if an adversary finds a failing ciphertext in less than Q_i samples, she can query more ciphertexts in the next stages $i+1, \ldots, N$. However, this effect is small due to the large value of Q_i.

The total amount of precomputation quantum work and the total amount of queries to obtain the N failing ciphertexts by performing Q_i tries for each ciphertext, can be expressed as

$$\mathcal{W}_N^{\text{tot}} := \sum_{i=0}^{N-1} \underbrace{\frac{Q_i}{\sqrt{\alpha_{i,f_i}}}}_{:=W_i} \qquad \mathcal{Q}_N^{\text{tot}} := \sum_{i=0}^{N-1} Q_i. \tag{4.22}$$

Recall that for now, we assume there is no upper limit to the number of decryption queries that can be made, and we focus on minimizing the amount of work. The values of Q_i that minimize the total quantum work $\mathcal{W}_N^{\text{tot}}$ can be found using the following Lagrange multiplier, minimizing the total amount of work to find N failures with probability $1-e^{-1}$ using the above probability model:

$$L(Q_0, \cdots, Q_{N-1}, \lambda) = \sum_{t=0}^{N-1} \frac{Q_i}{\sqrt{\alpha_{i,f_i}}} + \lambda\left((1 - e^{-1}) - \prod_{i=0}^{N-1}(1 - e^{-Q_i\beta_{i,f_i}})\right) \tag{4.23}$$

By equating the partial derivative of L in $Q_0, \cdots, Q_{N-1}$ and λ to zero and solving the resulting system of equations, we obtain the optimal values of $Q_0, \cdots, Q_{N-1}$ to mount our attack.

The resulting total work and queries of obtaining N ciphertext using directional failure boosting are given in Table 4.4 and Fig. 4.5. One can see that the majority of the work lies in obtaining the first ciphertext and that obtaining more than one ciphertext can be

ciphertexts N	1	2	3	5	10	20	30
$\log_2(\mathcal{W}_N^{\text{tot}})$	112.45	112.77	112.78	112.78	112.78	112.78	112.78
$\log_2(\mathcal{W}_N^{\text{tot}}/\mathcal{W}_1^{\text{tot}})$	—	0.32	0.33	0.33	0.33	0.33	0.33
$\log_2(\mathcal{Q}_N^{\text{tot}})$	102.21	102.86	102.87	102.87	102.87	102.87	102.87
$\log_2(\mathcal{Q}_N^{\text{tot}}/\mathcal{Q}_1^{\text{tot}})$	—	0.65	0.66	0.66	0.66	0.66	0.66

Table 4.4: *Quantum work $\mathcal{W}_N^{\text{tot}}$ and queries $\mathcal{Q}_N^{\text{tot}}$ required to find N failing ciphertexts with probability $1 - e^{-1}$. Finding the first ciphertext requires the heaviest amount of computation. After the third failing ciphertext is found, the following ones are essentially for free.*

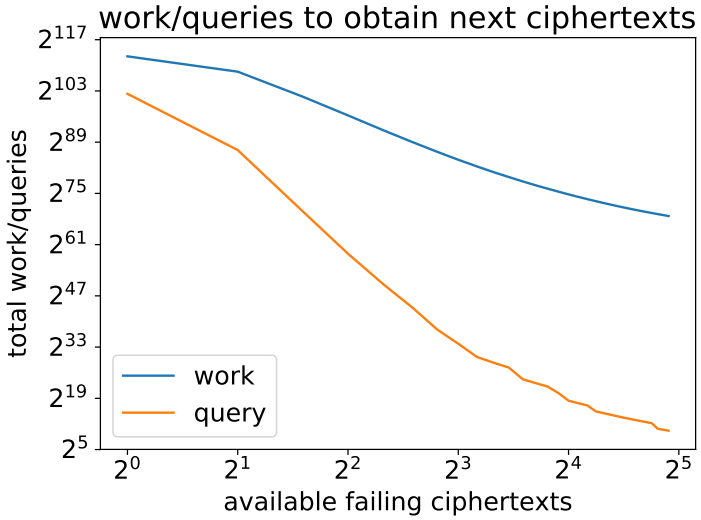


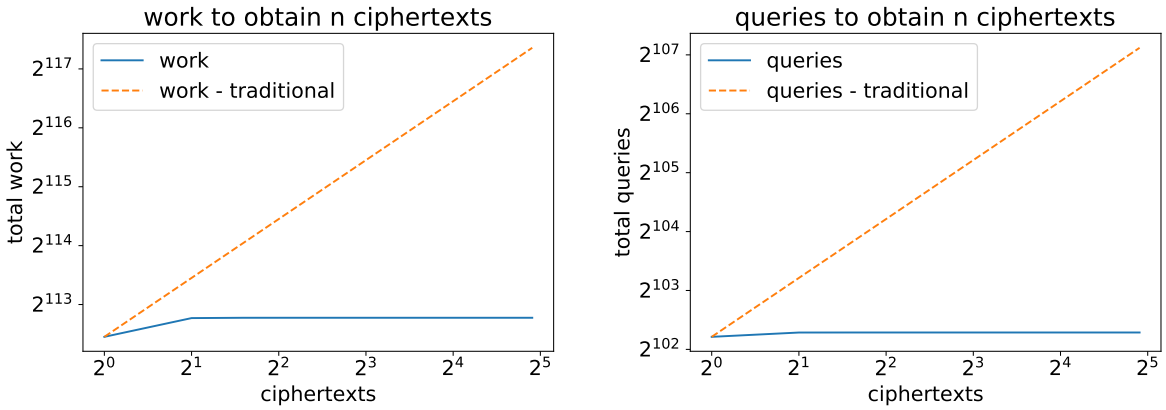
Figure 4.5: *Quantum work W_i and number of decryption queries Q_i required to find a new failing ciphertext, given the i failing ciphertexts found previously.*

done in less than double the work and queries, or less than one extra bit of complexity. For schemes with a lower failure probability, failing ciphertexts will be more correlated to the secret, so that the directional information is higher and directional failure boosting will be more effective.

In conclusion, the security of a scheme with low failure probability under a single target decryption failure attack can be approximated by the amount of work and queries that an adversary needs to do in order to obtain the first decryption failure. We emphasize the fact that obtaining many failures for a low overhead threatens the security of the scheme (See Section 4.3.3).

4.3.5 Applications

In Fig. 4.6, the total work and queries needed to obtain n ciphertexts with probability $1 - e^{-1}$ is plotted for both the traditional failure boosting, and our directional failure boosting approach. For a fair comparison between both results, we adapted the method for estimating the total work and queries with success probability $1 - e^{-1}$ using the traditional failure boosting of [DAn+19a]. For more information about our method, we refer to our paper [DRV20].



(a) Quantum work $\mathcal{W}_N^{\text{tot}}$ **(b)** Queries $\mathcal{Q}_N^{\text{tot}}$

Figure 4.6: *Quantum work $\mathcal{W}_N^{\text{tot}}$ and number of decryption queries $\mathcal{Q}_N^{\text{tot}}$ required to obtain N failing ciphertexts with probability $1 - e^{-1}$, given the number of previously found failing ciphertexts.*

	l	n	q	σ_s	σ_e	Failure probability	Claimed Security
ss-ntru-pke	1	1024	$2^{30} + 2^{13} + 1$	724	724	$> 2^{-80}$	2^{198}

Table 4.5: *Parameters of the ss-ntru-pke [Zha+17] scheme.*

Minimizing the number of queries instead

In case a maximal number of decryption queries is imposed, say $\mathtt{Q} = 2^K$, the same attack strategy can be applied. However, to limit the number of queries $\mathcal{Q}_N^{\text{tot}}$ necessary in the attack, a stronger preprocessing $\sqrt{\alpha_{i,f_t}^{-1}}$ might be necessary to increase the failure probability β_{i,f_t} of weak ciphertexts over 2^{-K}. The only change needed to accomplish this consists in selecting the threshold f_t for each i appropriately.

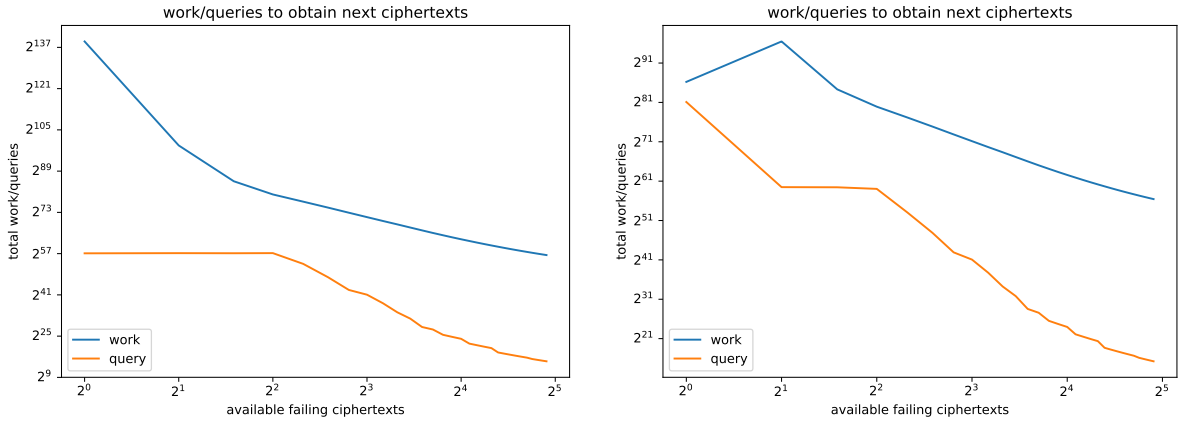
Too much preprocessing when $K = 64$. Note that, for most practical schemes (e.g. Kyber or Saber), increasing the failure probability β_{0,f_t} over 2^{-64} is not practically feasible or would require too much preprocessing $\sqrt{\alpha_{0,f_t}^{-1}}$. For example, considering the parameters given in Table 4.1, achieving $\beta_{0,f_t} > 2^{-64}$ requires an amount of total quantum work for finding the first failure larger than 2^{190}.

Application to ss-ntru-pke and improvement of Guo et al. [GJN19]

In [GJN19], an adaptive multitarget attack is proposed on the ss-ntru-pke version of NTRUEncrypt [Zha+17], a Ring-LWE based encryption scheme that claims security against chosen ciphertext attacks. The parameters of this scheme are given in Table 4.5.

The attack performs at most 2^{64} queries on at most 2^{64} targets and has a classical cost of 2^{216} work, and a quantum cost of 2^{140} when speeding up the offline phase with Grover's search. We adapt directional failure boosting to this attack model and propose both a single and multitarget attack.

For the single target attack, our proposed methodology in Section 4.3.4 needs more than 2^{64} queries to obtain a ciphertext. To mitigate this, we increase the pre-computational



(a) Single target attack **(b)** Multitarget attack

Figure 4.7: *Quantum work $\mathcal{W}_N^{\text{tot}}$ and number of decryption queries $\mathcal{Q}_N^{\text{tot}}$ required to find a new failing ciphertext for ss-ntru-pke, given the ones found previously.*

scheme	claimed security	multitarget attack [GJN]	our single target attack	our multitarget attack
ss-ntru-pke	2^{198}	$2^{139.5}$	$2^{139.6}$	$2^{96.6}$

Table 4.6: *Comparison of costs for different attacks against ss-ntru-pke [Zha+17].*

work $\sqrt{\alpha^{-1}}$ so that the failure probability of weak ciphertexts β increases over a certain f_t, which is chosen as 2^{-57} to make sure the total queries are below 2^{64}. The effect is a bigger overall computation, but a reduction in the number of necessary decryption queries. The rest of the attack proceeds as discussed in Section 4.3.4. The work or queries needed to obtain an extra failing ciphertext with the knowledge of N failing ciphertexts can be seen in Fig. 4.7a. The cost of this single target attack is $2^{139.6}$, which is close to the cost of their multitarget attack $2^{139.5}$, as can be seen in Table 4.6.

In the multitarget case, we can use a maximum of $2^{64} \cdot 2^{64}$ queries to find the first failing ciphertext, after which we use the methodology of the single target attack to obtain further ciphertext with a limited amount of queries. In practice, we stay well below the query limit to find the first failure. In this case, the work is dominated by finding the second decryption failure, as we need to do this in under 2^{64} queries. The resulting work to obtain an extra ciphertext is depicted in Fig. 4.7b. The cost of this attack is $2^{96.6}$, which is well below the cost of $2^{139.5}$ reported by Guo et al.

4.4 Perspectives

This chapter provides an intuition of the danger of the decryption failures for the lattice-based KEMs.

- When implemented in a misuse situation with key reuse in the IND-CPA version, the security of a lattice-based scheme can dramatically decrease. For the example of NewHope (Section 4.2), the security is largely decreased with less than 2^{14} queries; and it can be totally broken with less than 2^{20} queries. It is thus absolutely necessary to avoid these misuse situations.

- When implemented in an IND-CCA version, the work of Section 4.3 leads us to believe that having access to one failure could jeopardize the whole security. However, our results lays the first stones, and further work is still required. First, our results only apply to a simple scheme that does not correspond to any of the NIST lattice-based candidates thought. Second, in our example, accessing to the first failure cost 2^{102} queries, which seem above the NIST limit of 2^{64}. Studying the security in a limited query setting (and/or in a multitarget case) is left as an interesting future work.

Perspective 5 (Application to schemes with error correction)**.** *For schemes with strong error correction, such as LAC [Lu+19], the error probability of the ciphertext before error correction is relatively high, which results in a lower correlation between the secret and failing ciphertexts. This would complicate the effectiveness of directional failure boosting in several ways: finding the right rotation of ciphertext to combine them into* $\overline{\mathbf{E}}$ *becomes less straightforward, and the resulting* $\overline{\mathbf{E}}$ *would be less correlated with the secret. However, by finding one failure, an adversary would already have more than one equation due to the multiple errors. Thus the effectiveness of directional failure boosting in these circumstances can not be derived from our results and would be an interesting future research topic.*

Perspective 6 (Multitarget attack with limited number of queries.)**.** *In case an attacker has multiple possible targets* 2^T *but can perform at most* 2^K *queries at each target, she can initially search for one of more ciphertexts using a small and thus more optimal* $\beta_{f_t} > 2^{-(T+K)}$*, before focussing on one target and limiting to* $\beta_{f_t} > 2^{-K}$*, which requires more preprocessing and possibly more work. In this case, finding the first decryption failure is not necessarily the limiting factor for the adversary. While the attack strategy has been touched upon in Section 4.3.5, finding the optimal attack strategy and its efficiency would be interesting future work.*

Perspective 7 (On the choice of the 2^{64} queries limit)**.** *As seen in Chapters 2 and 4, the queries limitation is crucial for security assessments. One the one hand, it helps defining the parameters to avoid timing and classical attacks. On the other hand, it helps avoiding decryption failure attacks. All the security estimations are based on the* 2^{64} *queries limit suggested by the NIST. Allowing more queries, in an order of* 2^{80} *for instance, would be devastating for existing schemes. For ensuring the same security, the designers would need to derive larger parameters and, thus, slower algorithms. Besides, the* 2^{64} *limits seems low for schemes that are supposed to be instantiated in the next decades. An interesting perspective can be to precisely study and motivate a choice for the maximum queries limit depending of the future usage in the cryptographic protocols.*

Chapter

5 Lattice reduction attacks with side information

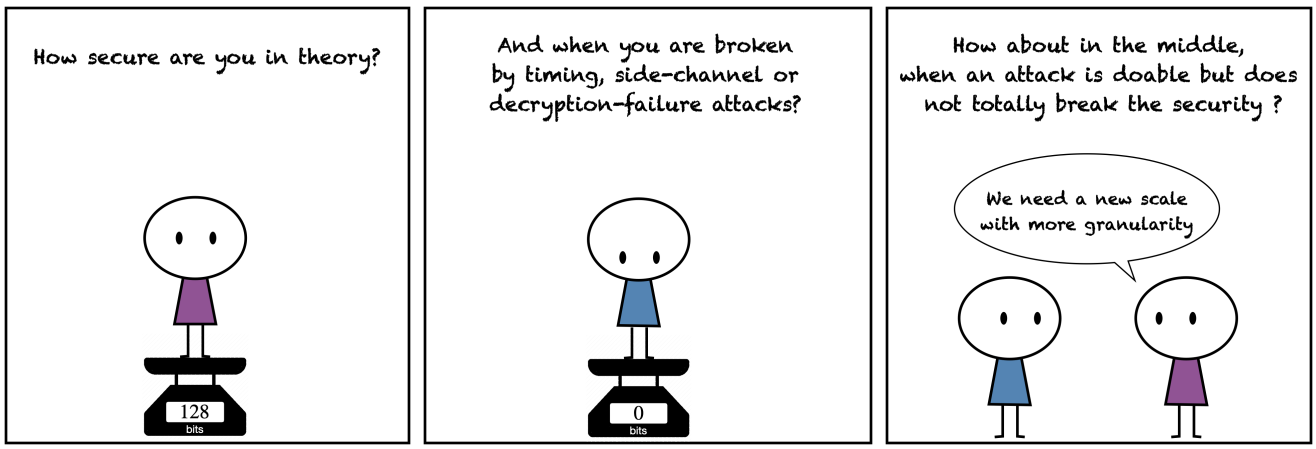


Figure 5.1: *The security of an algorithm is commonly expressed in "bits", where n-bit security means that the attacker would have to perform 2^n operations to break it. After a complete timing, physical or decryption-failure attack, the full secret is known and the security decreases to 0. However, the attacks do not often provide the full secret but only side information on it. This chapter gives a way to compute the remaining attack complexity when some side information on the secret is given.*

This work has been the last project of this PhD. It somehow wraps up the works of the previous chapters in a unified framework to estimate the complexity of lattice reduction with side information (side-channels, failures or implementation constraints). Huijing Gong and her advisor Dana Dachman-Soled joined Léo Ducas and me in this project and they provided precious help on the theoretical side.

This chapter corresponds to the final version of [Dac+20] recently accepted to the CRYPTO conference[1]. It has been very interesting and stimulating to develop such a federating framework with so many applications. Although my contributions are mainly related to the inclusion of the hints and the applications along with their implementations, everyone of us contributed to all parts of this research.

The tool that we designed looks very handy for cryptanalysis and, hopefully, it will help finish attacks that recover partial information on the secret. To ensure an easy use, a small tutorial is presented on the Github of our tool at: https://github.com/lducas/leaky-LWE-Estimator.

[1] We refer to our animated Crypto video https://www.youtube.com/watch?v=wCaLcbWnwDI for a presentation of the geometric intuition.

Chapter content

5.1 Introduction and motivations

The ongoing standardization process and anticipated deployment of lattice-based cryptography raises an important question: *How resilient are lattices to side-channel attacks or other forms of side information?* While there are numerous works addressing this question for specific cryptosystems (See [ADP18b; Bru+16; GP18; Rav+19; Rav+18; Bos+18a] for side-channel attacks targeting lattice-based NIST candidates), these works use rather ad-hoc methods to reconstruct the private key, requiring new techniques and algorithms to be developed for each setting. For example, the work of [Bos+18a] uses brute-force methods for a portion of the attack, while [Boo+18] exploits linear regression techniques. Moreover, ad-hoc methods do not allow (1) to take advantage of decades worth of research and (2) optimization of standard lattice attacks. Second, most of the side-channel attacks from prior work consider substantial amounts of information leakage and show that it leads to feasible recovery of the entire key, whereas one may be interested in more precise tradeoffs in terms of information leakage versus concrete security of the scheme. The above motivates the focus of this chapter: Can one integrate side information into a standard lattice attack, and if so, by how much does the information reduce the cost of this attack? Given that side-channel resistance is the next step toward

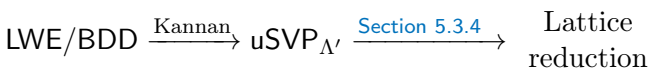


Figure 5.2: *Primal attack without hints (prior art).*

the technological readiness of lattice-based cryptography, and that we expect numerous new projects in this growing area, we believe that a general framework and a prediction software are in order.

In this chapter, we propose a framework that generalizes the so-called primal lattice reduction attack, and allows the progressive integration of "hints" (i.e. side information that takes one of several forms) before running the final lattice reduction step. This study is summarized in Figs. 5.2 and 5.3 and developed in Section 5.3.We also present a Sage 9.0 toolkit to actually mount such attacks with hints when computationally feasible, and to predict their performance on larger instances. Our predictions are validated by extensive experiments. Our tool and these experiments are described in Section 5.5. We finally demonstrate the usefulness of our framework and tool via three example applications. Our main example (Section 5.6.1) revisits the side-channel information obtained from the first side-channel attack of [Bos+18a] against Frodo. In that article, it was concluded that a divide-and-conquer side-channel template attack would not lead to a meaningful attack using standard combinatorial search for reconstruction of the secret. Our technique allows to integrate this side-channel information into lattice attacks, and to predict the exact security drop. For example, the CCS2 parameter set very conservatively aims for 128-bits of post-quantum security (or 448 "bikz" as defined in Section 5.3.4); but after the leakage of [Bos+18a] we predict that its security drops to 29 "bikz", i.e. that it can be broken with BKZ-29.

Interestingly, we note that our framework is not only useful in the side-channel scenario; we are, for example, also able to model decryption failures as hints fitting our framework. This allows us to reproduce some predictions from [DVV18]. This is discussed in Section 5.6.2.

Perhaps more surprisingly, we also find a novel improvement to analyze a few schemes (LAC [Lu+19], Round5 [Gar+19]) without any side-channel or oracle queries. Indeed, such schemes use ternary distribution for secrets, with a prescribed numbers of 1 and -1: this hint fits our framework, and lead to a (very) minor improvement, discussed in Section 5.6.3.

Lastly, our framework also encompasses and streamlines existing tweaks of the primal attack: the choice of ignoring certain LWE equations to optimize the volume-dimension trade-off, as well as the re-centering [Ngu19] and isotropization [Hof+09; Che+18] accounting for potential a-priori distortions of the secret. It also implicitly solves the question of the optimal choice of the coefficient for Kannan Bounded Distance Decoding problem (denoted BDD see Hard Problem 2) to unique Shortest Vector Problem (denoted uSVP see Hard Problem 1) [Kan87].

Technical overview. Our work is based on a generalization of the Bounded Distance Decoding problem (see Hard Problem 2) to a Distorted version (DBDD), which allows to account for the potentially non-spherical covariance of the secret vector to be found.

Each hint will affect the lattice itself, the mean and/or the covariance parameter of the DBDD instance, making the problem easier (see Fig. 5.3). At last, we make the distribution spherical again by applying a well-chosen linear transformation, reverting to a spherical BDD instance before running the attack. Thanks to the hints, this new

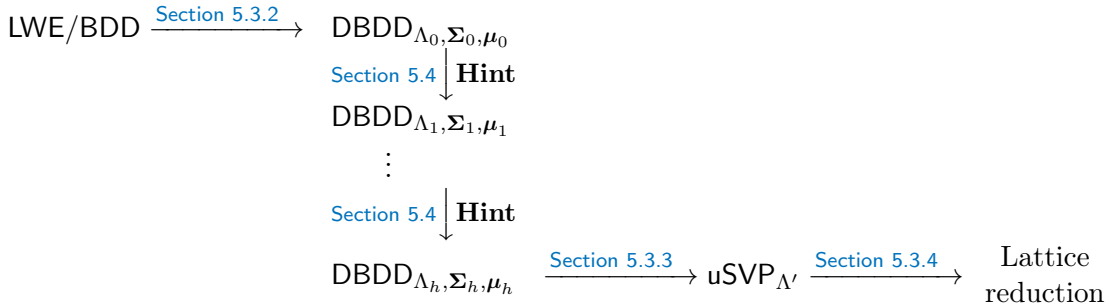


Figure 5.3: *The primal attack with hints (our work).*

instance will be easier than the initial one. Let us assume that $\mathbf{v}$, l, k and σ are parameters known by the attacker. Our framework can handle four types of hints on the secret $\mathbf{s}$ or on the lattice Λ.

- Perfect hints: $\langle \mathbf{s},\ \mathbf{v} \rangle = l$ — *intersect the lattice with an hyperplane.*
- Modular hints : $\langle \mathbf{s},\ \mathbf{v} \rangle = l \bmod k$ — *sparsify the lattice.*
- Approximate hints : $\langle \mathbf{s},\ \mathbf{v} \rangle = l + \epsilon_\sigma$ — *decrease the covariance of the secret.*
- Short vector hints : $\mathbf{v} \in \Lambda$ — *project orthogonally to* $\mathbf{v}$.

While the first three hints are clear wins for the performance of lattice attacks, the last one is a trade-off between the dimension and the volume of the lattice. This last type of hint is in fact meant to generalize the standard trick consisting of 'ignoring' certain LWE equations; ignoring such an equation can be interpreted geometrically as such a projection orthogonally to a so-called q-vector.

All the transformations of the lattice above can be computed in polynomial time. However, computing with general distribution in large dimension is not possible; we restrict our study to the case of Gaussian distributions of arbitrary covariance, for which such computations are also poly-time.

Some of these transformations remain quite expensive, in particular, because they involve rational numbers with very large denominators, and it remains rather impractical to run them on cryptographic-grade instances. Fortunately, up to a necessary hypothesis of primitivity of the vector $\mathbf{v}$ (with respect to either Λ or its dual depending on the type of hint), we can also predict the effect of each hint on the lattice parameters, and therefore run faster predictions of the attack cost.

From Leaks to Hints. At first, it may not be so clear that the types of hints above are so useful in realistic applications, in particular since they need to be linear on the secret. Of course our framework can handle rather trivial hints such as the perfect leak of a secret coefficient $\mathbf{s}_i = l$. Slightly less trivial is the case where only the low-order bits leaks, a hint of the form $\mathbf{s}_i = l \bmod 2$.

We note that most of the computations done during an LWE decryption are linear: leaking any intermediate register during a matrix vector product leads to a hint of the same form (possibly $\bmod q$). Similarly, the leak of a NTT coefficient of a secret in a Ring/Module variant can also be viewed as such.

Admittedly, such ideal leaks of a full register are not the typical scenario and leaks are typically not linear (for the addition in $\mathbb{Z}_q$) on the content of the register. However, such non-linearities can be handled by approximate hints. For instance, let $\mathbf{s}_0$ be a

secret coefficient (represented by a signed 16-bits integer), whose a priori distribution is supported by $\{-5, \ldots, 5\}$. Consider the case where we learn the Hamming weight of $\mathbf{s}_0$, say $H(\mathbf{s}_0) = 2$. Then, we can narrow down the possibilities to $\mathbf{s}_0 \in \{3, 5\}$. This leads to two hints:

- a modular hint: $\mathbf{s}_0 = 1 \bmod 2$,
- an approximate hint: $\mathbf{s}_0 = 4 + \epsilon_1$, where ϵ_1 has variance 1.

While closer to a realistic scenario, the above example remains rather simplified. A detailed example of how realistic leaks can be integrated as hints will be given in Section 5.6.1, based on the leakage data from [Bos+18a].

5.2 Preliminaries

5.2.1 Linear Algebra

We use bold lower case letters to denote vectors, and bold upper case letters to denote matrices. Let $\langle \cdot, \cdot \rangle$ denote the inner product of two vectors of the same size. Let us introduce the row span of a matrix (denoted $\mathrm{Span}(\cdot)$) as the subspace generated by all $\mathbb{R}$-linear combinations of the rows of its input.

Definition 21 (Positive Semidefinite)**.** *A $n \times n$ symmetric real matrix $\mathbf{M}$ is positive semidefinite if scalar $\mathbf{x}\mathbf{M}\mathbf{x}^T \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$; if so we write $\mathbf{M} \geq 0$. Given two $n \times n$ real matrix $\mathbf{A}$ and $\mathbf{B}$, we note $\mathbf{A} \geq \mathbf{B}$ if $\mathbf{A} - \mathbf{B}$ is positive semidefinite.*

Definition 22. *A matrix $\mathbf{M}$ is a square root of $\mathbf{\Sigma}$, denoted $\sqrt{\mathbf{\Sigma}}$, if*

$$\mathbf{M}^T \cdot \mathbf{M} = \mathbf{\Sigma},$$

Our techniques involve keeping track of the covariance matrix $\mathbf{\Sigma}$ of the secret and error vectors as hints are progressively integrated. The covariance matrix may become singular during this process and will not have an inverse. Therefore, in the following, we introduce some degenerate notions for the inverse and the determinant of a square matrix. Essentially, we restrict these notions to the row span (denoted $\mathrm{Span}()$) of their input. For $\mathbf{X} \in \mathbb{R}^{d \times k}$ (with any $d, k \in \mathbb{N}$), we will denote $\mathbf{\Pi}_\mathbf{X}$ the orthogonal projection matrix onto $\mathrm{Span}(\mathbf{X})$. More formally, let $\mathbf{Y}$ be a maximal set of independent row-vectors of $\mathbf{X}$; the orthogonal projection matrix is given by $\mathbf{\Pi}_\mathbf{X} \coloneqq \mathbf{Y}^T \cdot (\mathbf{Y} \cdot \mathbf{Y}^T)^{-1} \cdot \mathbf{Y}$. Its complement (the projection orthogonally to $\mathrm{Span}(\mathbf{X})$) is denoted $\mathbf{\Pi}_\mathbf{X}^\perp \coloneqq \mathbf{I}_d - \mathbf{\Pi}_\mathbf{X}$. We naturally extend the notation $\mathbf{\Pi}_F$ and $\mathbf{\Pi}_F^\perp$ to subspaces $F \subset \mathbb{R}^d$. By definition, the projection matrices satisfy $\mathbf{\Pi}_F^2 = \mathbf{\Pi}_F$, $\mathbf{\Pi}_F^T = \mathbf{\Pi}_F$ and $\mathbf{\Pi}_F \cdot \mathbf{\Pi}_F^\perp = \mathbf{\Pi}_F^\perp \cdot \mathbf{\Pi}_F = \mathbf{0}$.

Definition 23 (Restricted inverse and determinant)**.** *Let $\mathbf{\Sigma}$ be a symmetric matrix. We define a* restricted inverse *denoted $\mathbf{\Sigma}^\sim$ as*

$$\mathbf{\Sigma}^\sim \coloneqq (\mathbf{\Sigma} + \mathbf{\Pi}_\mathbf{\Sigma}^\perp)^{-1} - \mathbf{\Pi}_\mathbf{\Sigma}^\perp.$$

It satisfies $\mathrm{Span}(\mathbf{\Sigma}^\sim) = \mathrm{Span}(\mathbf{\Sigma})$ and $\mathbf{\Sigma} \cdot \mathbf{\Sigma}^\sim = \mathbf{\Pi}_\mathbf{\Sigma}$.

We also denote $\mathrm{rdet}(\mathbf{\Sigma})$ as the restricted determinant *defined as follows.*

$$\mathrm{rdet}(\mathbf{\Sigma}) \coloneqq \det(\mathbf{\Sigma} + \mathbf{\Pi}_\mathbf{\Sigma}^\perp).$$

The idea behind Definition 23 is to provide an (artificial) invertibility property to the input $\mathbf{\Sigma}$ by adding the missing orthogonal part and to remove it afterwards. For example, if $\mathbf{\Sigma} = \begin{bmatrix} \mathbf{A} & 0 \\ 0 & 0 \end{bmatrix}$ where $\mathbf{A}$ is invertible,

$$\mathbf{\Sigma}^{\sim} = \left(\begin{bmatrix} \mathbf{A} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} - \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \operatorname{rdet} \mathbf{\Sigma} = \det(\mathbf{A}).$$

5.2.2 Statistics

Recall that random variables, i.e. variables whose values depend on outcomes of a random phenomenon, are denoted in lowercase calligraphic letters e.g. $\mathfrak{a}, \mathfrak{b}$. Random vectors are denoted in uppercase calligraphic letters e.g. $\mathfrak{C}, \mathfrak{X}, \mathfrak{Z}$.

Before hints are integrated, we will assume that the secret and error vectors follow a multidimensional normal (Gaussian) distribution. Hints will typically correspond to learning a (noisy, modular or perfect) linear equation on the secret. We must then consider the altered distribution on the secret, conditioned on this information. Fortunately, this will also be a multidimensional normal distribution with an altered covariance and mean. In the following, we present the precise formulae for the covariance and mean of these conditional distributions. The notations derive from the ones defined Section 2.2.2 because we consider a generalized continuous distribution.

Definition 24 (Multidimensional normal distribution)**.** *Let $d \in \mathbb{Z}$, for $\boldsymbol{\mu} \in \mathbb{Z}^d$ and $\mathbf{\Sigma}$ being a symmetric positive semidefinite matrix of dimension $d \times d$, we denote by $G^d_{\mathbf{\Sigma},\boldsymbol{\mu}}$ the multidimensional normal distribution supported by $\boldsymbol{\mu} + \operatorname{Span}(\mathbf{\Sigma})$ by the following*

$$\mathbf{x} \in \mathbb{R}_d \mapsto \frac{1}{\sqrt{(2\pi)^{\operatorname{rank}(\mathbf{\Sigma})} \cdot \operatorname{rdet}(\mathbf{\Sigma})}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}) \cdot \mathbf{\Sigma}^{\sim} \cdot (\mathbf{x} - \boldsymbol{\mu})^T \right).$$

Theorem 12 shows the altered distribution of a normal random variable conditioned on its noisy linear transformation value, following from the more generalized result of [Liu19, Equations (6) and (7)], which we refer to for the calculative proof.

Theorem 12 (Conditional distribution $\mathfrak{X} | \langle \mathfrak{X}, \mathbf{v} \rangle + \mathfrak{e}$)**.** *Suppose that $\mathfrak{X} \in \mathbb{Z}^d$ has a $G^d_{\mathbf{\Sigma},\boldsymbol{\mu}}$ distribution and $\mathfrak{e}$ has a $G^1_{\sigma_{\mathfrak{e}}^2,0}$ distribution. Let us fix $\mathbf{v} \in \mathbb{R}^d$ as a nonzero vector and $z \in \mathbb{Z}$. We define the following scalars:*

$$\mathfrak{z} \coloneqq \langle \mathfrak{X}, \mathbf{v} \rangle + \mathfrak{e},\ \mu_2 \coloneqq \langle \mathbf{v}, \boldsymbol{\mu} \rangle \text{ and } \sigma_2 \coloneqq \mathbf{v} \mathbf{\Sigma} \mathbf{v}^T + \sigma_{\mathfrak{e}}^2$$

If $\sigma_2 \neq 0$, the conditional distribution of $\mathfrak{X} \Big| (\mathfrak{z} = z)$ is $G^d_{\mathbf{\Sigma}',\boldsymbol{\mu}'}$, where

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \frac{(z - \mu_2)}{\sigma_2} \mathbf{v} \mathbf{\Sigma}$$
$$\mathbf{\Sigma}' = \mathbf{\Sigma} - \frac{\mathbf{\Sigma} \mathbf{v}^T \mathbf{v} \mathbf{\Sigma}}{\sigma_2}.$$

If $\sigma_2 = 0$, the conditional distribution of $\mathfrak{X} \Big| (\mathfrak{z} = z)$ is $G^d_{\mathbf{\Sigma},\boldsymbol{\mu}}$.

Remark 20. *We note that Theorem 12 is also valid for $\mathfrak{X} | \langle \mathfrak{X}, \mathbf{v} \rangle$ by letting $\sigma_{\mathfrak{e}} = 0$.*

5.2.3 Lattices

Recall for Section 5.1 that a *lattice*, denoted here as Λ, is a discrete additive subgroup of $\mathbb{R}^m$, which is generated as the set of all linear integer combinations of n ($m \geq n$) linearly independent basis vectors $\{\mathbf{b}_j\} \subset \mathbb{R}^m$, namely,

$$\Lambda \coloneqq \left\{ \sum\nolimits_j z_j \mathbf{b}_j : z_j \in \mathbb{Z} \right\},$$

A matrix $\mathbf{B}$ having the basis vectors as rows is called a *basis* matrix. Recall that the *volume* of a lattice Λ is defined as $\mathrm{Vol}(\Lambda) \coloneqq \sqrt{\det(\mathbf{B}\mathbf{B}^T)}$. The *dual lattice* of Λ in $\mathbb{R}^n$ is defined as follows.

$$\Lambda^* \coloneqq \{\mathbf{y} \in \mathrm{Span}(\mathbf{B}) \mid \forall \mathbf{x} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}.$$

Note that, $(\Lambda^*)^* = \Lambda$, and $\mathrm{Vol}(\Lambda^*) = 1/\mathrm{Vol}(\Lambda)$. For a detailed introduction of dual lattices properties, we refer to [Mic20].

Lemma 18 ([Mar13, Proposition 1.3.4])**.** *Let Λ be a lattice and let F be a subspace of $\mathbb{R}^n$. If $\Lambda \cap F$ is a lattice, then the dual of $\Lambda \cap F$ is the orthogonal projection onto F of the dual of Λ. In other words, each element of Λ^* is multiplied by the projection matrix $\mathbf{\Pi}_F$:*

$$(\Lambda \cap F)^* = \Lambda^* \cdot \mathbf{\Pi}_F.$$

Definition 25 (Primitive vectors)**.** *A set of vector $\mathbf{y}_1, \ldots, \mathbf{y}_k \in \Lambda$ is said primitive with respect to Λ if $\Lambda \cap \mathrm{Span}(\mathbf{y}_1, \ldots, \mathbf{y}_k)$ is equal to the lattice generated by $\mathbf{y}_1, \ldots, \mathbf{y}_k$. Equivalently, it is primitive if it can be extended to a basis of Λ. If $k = 1$, $\mathbf{y}_1$, this is equivalent to $\mathbf{y}_1/i \notin \Lambda$ for any integer $i \geq 2$.*

To predict the hardness of the lattice reduction on altered instances , we must compute the volume of the final transformed lattice. We devise a highly efficient way to do this, by observing that each time a hint is integrated, we can update the volume of the transformed lattice, given only the volume of the previous lattice and information about the current hint (under mild restrictions on the form of the hint). In the following, we present technical lemmas that will be useful for progressively computing the volume of the transformed lattice when different types of hints are integrated. We refer to the original paper for their proofs [Dac+20].

Lemma 19 (Volume of a lattice slice)**.** *Given a lattice Λ with volume $\mathrm{Vol}(\Lambda)$, and a primitive vector $\mathbf{v}$ with respect to Λ^*. Let $\mathbf{v}^\perp$ denote subspace orthogonal to $\mathbf{v}$. Then $\Lambda \cap \mathbf{v}^\perp$ is a lattice with volume $\mathrm{Vol}(\Lambda \cap \mathbf{v}^\perp) = \|\mathbf{v}\| \cdot \mathrm{Vol}(\Lambda)$.*

Lemma 20 (Volume of a sparsified lattice)**.** *Let Λ be a lattice, $\bar{\mathbf{v}} \in \Lambda^*$ be a primitive vector of Λ^*, and $k > 0$ be an integer. Let $\Lambda' = \{\mathbf{x} \in \Lambda \mid \langle \mathbf{x}, \bar{\mathbf{v}} \rangle = 0 \bmod k\}$ be a sublattice of Λ. Then $\mathrm{Vol}(\Lambda') = k \cdot \mathrm{Vol}(\Lambda)$.*

Lemma 21 (Volume of a projected lattice)**.** *Let Λ be a lattice, $\mathbf{v} \in \Lambda$ be a primitive vector of Λ. Let $\Lambda' = \Lambda \cdot \mathbf{\Pi}_{\mathbf{v}}^{\perp}$ be a sublattice of Λ. Then $\mathrm{Vol}(\Lambda') = \mathrm{Vol}(\Lambda)/\|\mathbf{v}\|$. More generally, if $\mathbf{V}$ is a primitive set of vectors of Λ, then $\Lambda' = \Lambda \cdot \mathbf{\Pi}_{\mathbf{V}}^{\perp}$ has volume $\mathrm{Vol}(\Lambda') = \mathrm{Vol}(\Lambda)/\sqrt{\det(\mathbf{V}\mathbf{V}^T)}$.*

Lemma 22 (Lattice volume under linear transformations)**.** *Let Λ be a lattice in $\mathbb{R}^n$, and $\mathbf{M} \in \mathbb{R}^{n \times n}$ a matrix such that $\ker \mathbf{M} = \mathrm{Span}(\Lambda)^\perp$. Then we have $\mathrm{Vol}(\Lambda \cdot \mathbf{M}) = \mathrm{rdet}(\mathbf{M})\,\mathrm{Vol}(\Lambda)$.*

5.3 Distorted Bounded Distance Decoding (DBDD)

5.3.1 Definition

We first introduce a variant of the LWE problem as presented in Hard Problem 3. Indeed, we need the search LWE problem in its short-secret variant which is the most relevant to practical LWE-based encryption. As shown in [Alb+18], all the (Mod, Ring, plain)LWE-based schemes can be embedded into Hard Problem 10.

Hard Problem 10 — Search LWE problem with short secrets

Let n, m and q be positive integers, and let χ be a distribution over $\mathbb{Z}$.

Given the pair $\left(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{b} = \mathbf{z}\mathbf{A}^T + \mathbf{e} \in \mathbb{Z}_q^m\right)$ where:

1. $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is sampled uniformly at random,
2. $\mathbf{z} \leftarrow \chi^n$, and $\mathbf{e} \leftarrow \chi^m$ are sampled with independent and identically distributed coefficients following the distribution χ.

Find $\mathbf{z}$.

The primal attack (See for example [Alb+17]) against (search)-LWE proceeds by viewing the LWE instance as an instance of a Bounded Distance Decoding (BDD) problem, converting it to a uSVP instance (via Kannan's embedding [Kan87]), and finally applying a lattice reduction algorithm to solve the uSVP instance. The central tool of our framework is a generalization of BDD that accounts for potential distortion in the distribution of the secret noise vector that is to be recovered.

Hard Problem 11 — Distorted Bounded Distance Decoding problem (DBDD)

Let $\Lambda \subset \mathbb{R}^d$ be a lattice, $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ be a symmetric matrix and $\boldsymbol{\mu} \in \mathrm{Span}(\Lambda) \subset \mathbb{R}^d$ such that

$$\mathrm{Span}(\mathbf{\Sigma}) \subsetneq \mathrm{Span}(\mathbf{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \mathrm{Span}(\Lambda). \tag{5.1}$$

The Distorted Bounded Distance Decoding problem $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\mathbf{\Sigma}}$ is the following problem:

Given $\boldsymbol{\mu}, \mathbf{\Sigma}$ and a basis of Λ.

Find the unique vector $\mathbf{x} \in \Lambda \cap E(\boldsymbol{\mu}, \mathbf{\Sigma})$

where $E(\boldsymbol{\mu}, \mathbf{\Sigma})$ denotes the ellipsoid

$$E(\boldsymbol{\mu}, \mathbf{\Sigma}) \coloneqq \{\mathbf{x} \in \boldsymbol{\mu} + \mathrm{Span}(\mathbf{\Sigma}) | (\mathbf{x} - \boldsymbol{\mu}) \cdot \mathbf{\Sigma}^{\sim} \cdot (\mathbf{x} - \boldsymbol{\mu})^T \leq \mathrm{rank}(\mathbf{\Sigma})\}.$$

We will refer to the triple $\mathcal{I} = (\Lambda, \boldsymbol{\mu}, \mathbf{\Sigma})$ as the instance of the $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\mathbf{\Sigma}}$ problem.

Intuitively, Hard Problem 11 corresponds to knowing that the secret vector $\mathbf{x}$ to be recovered follows a distribution of variance $\mathbf{\Sigma}$ and average $\boldsymbol{\mu}$. The quantity $(\mathbf{x} - \boldsymbol{\mu}) \cdot \mathbf{\Sigma}^{\sim} \cdot (\mathbf{x} - \boldsymbol{\mu})^T$ can be interpreted as a non-canonical Euclidean squared distance $\|\mathbf{x} - \boldsymbol{\mu}\|_{\mathbf{\Sigma}}^2$, and the expected value of such a distance for a Gaussian $\mathbf{x}$ of variance $\mathbf{\Sigma}$ and average $\boldsymbol{\mu}$ is $\mathrm{rank}(\mathbf{\Sigma})$. One can argue that, for such a Gaussian, there is a constant probability that $\|\mathbf{x} - \boldsymbol{\mu}\|_{\mathbf{\Sigma}}^2$ is slightly greater than $\mathrm{rank}(\mathbf{\Sigma})$[2]. Since we are interested in the average behavior of our attack, we ignore this benign technical detail. In fact, we will typically

[2]The choice of $\mathrm{rank}(\mathbf{\Sigma})$ as a radius is somewhat arbitrary and can be multiplied by any constant factor.

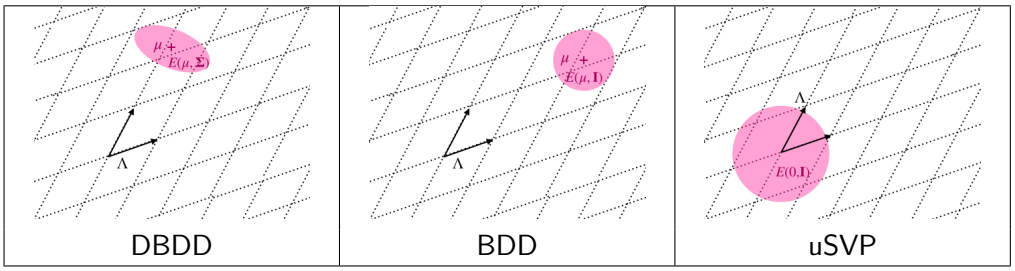


Figure 5.4: *Graphical intuition of DBDD, BDD and uSVP in dimension two: the problem consists in finding a nonzero element of Λ in the colored zone. The identity hyperball is larger for uSVP to represent the fact that, during the reduction, the uSVP lattice has one dimension more than for BDD.*

interpret DBDD as the promise that the secret follows a Gaussian distribution of center $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

The ellipsoid can be seen as an affine transformation (that we call "distortion") of the centered hyperball of radius $\text{rank}(\boldsymbol{\Sigma})$. Let us re-use the notation for the hyperball introduced in Eq. (1.1), i.e. for any $d \in \mathbb{N}$ $B_d(0, d)$ denotes an hyperball of dimension and radius d. In particular $B_d(0, d) = E(0, I_d)$. One can thus write using Definition 22:

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = B_{\text{rank}(\boldsymbol{\Sigma})}(0, \text{rank}(\boldsymbol{\Sigma})) \cdot \sqrt{\boldsymbol{\Sigma}} + \boldsymbol{\mu}. \tag{5.2}$$

From the Span inclusion in Eq. (5.1), one can deduce that the condition is equivalent to requiring $\boldsymbol{\mu} \notin \text{Span}(\boldsymbol{\Sigma})$ and $\text{rank}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \text{rank}(\boldsymbol{\Sigma}) + 1 = \text{rank}(\Lambda)$. This detail is necessary for embedding it properly into a uSVP instance (See later in Section 5.3.3).

Particular cases of Hard Problem 11. Let us temporarily ignore the condition in Eq. (5.1) to study some particular cases. As shown in Fig. 5.4, when $\boldsymbol{\Sigma} = \mathbf{I}_d$, $\text{DBDD}_{\Lambda,\boldsymbol{\mu},\mathbf{I}_d}$ is BDD instance. Indeed, the ellipsoid becomes a shifted hyperball $E(\boldsymbol{\mu}, \mathbf{I}_d) = \{\mathbf{x} \in \boldsymbol{\mu} + \mathbb{R}^{d\times d} \mid \|\mathbf{x} - \boldsymbol{\mu}\|_2 \leq d\} = B_d + \boldsymbol{\mu}$. If in addition $\boldsymbol{\mu} = 0$, $\text{DBDD}_{\Lambda,\mathbf{0},\mathbf{I}_d}$ becomes a uSVP instance on Λ.

5.3.2 Embedding LWE into DBDD

In the typical primal attack framework (Fig. 5.2), one directly views LWE as a BDD instance of the same dimension. For our purposes, however, it will be useful to apply Kannan's Embedding at this stage and therefore increase the dimension of the lattice by 1. While it could be delayed to the last stage of our attack, this extra fixed coefficient 1 will be particularly convenient when we integrate hints (see Remark 25 in Section 5.4). It should be noted that no information is lost through this transformation. Indeed, the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ allow us to encode the knowledge that the solution we are looking for has its last coefficient set to 1 and nothing else.

In more details, the solution $\mathbf{s} \coloneqq (\mathbf{e}, \mathbf{z})$ of an LWE instance is extended to

$$\bar{\mathbf{s}} \coloneqq (\mathbf{e}, \mathbf{z}, 1), \tag{5.3}$$

which is a short vector in the lattice $\Lambda = \left\{(\mathbf{x}, \mathbf{y}, w) \,|\mathbf{x} + \mathbf{y}\mathbf{A}^{\mathrm{T}} - \mathbf{b}w = 0 \mod q\right\}$. A basis of this lattice is given by the row vectors of

$$\begin{bmatrix} q\mathbf{I}_m & 0 & 0 \\ \mathbf{A}^{\mathrm{T}} & -\mathbf{I}_n & 0 \\ \mathbf{b} & 0 & 1 \end{bmatrix}.$$

Denoting μ_χ and σ_χ^2 the average and variance of the LWE distribution χ (See Hard Problem 10), we can convert this LWE instance to a $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ instance with $\boldsymbol{\mu} = [\mu_\chi \cdots \mu_\chi\ 1]$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_\chi^2 \mathbf{I}_{m+n} & 0 \\ 0 & 0 \end{bmatrix}$. The lattice Λ is of full rank in $\mathbb{R}^d$ where $d := m + n + 1$, and its volume is q^m. Note that the rank of $\boldsymbol{\Sigma}$ is only $d - 1$: the ellipsoid has one less dimension than the lattice. It then validates the requirement of Eq. (5.1).

Remark 21. *Typically, Kannan's embedding from BDD to uSVP leaves the bottom right matrix coefficient as a free parameter, say c, to be chosen optimally. The optimal value is the one maximizing*

$$\frac{\|(\mathbf{z}; c)\|}{\det(\Lambda)^{1/d}} = \frac{(m+n)\sigma_\chi + c}{(c \cdot q^m)^{1/d}},$$

namely, $c = \sigma_\chi$ according to the arithmetic-geometric mean inequality. Some prior work [Alb+17; BMW19] instead chose $c = 1$. While this is benign since σ_χ is typically not too far from 1, it remains a sub-optimal choice. Looking ahead, in our DBDD framework, this choice becomes irrelevant thanks to the isotropization *step introduced in the next section; we can therefore choose any value without affecting the final result.*

5.3.3 Converting DBDD to uSVP

In this section, we explain how a DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is converted into a uSVP one. Two modifications are necessary.

First, let us show that the ellipsoid in Hard Problem 11 is contained in a larger centered ellipsoid (with one more dimension) as follows:

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \subset E(\mathbf{0}, \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}). \tag{5.4}$$

Using Eq. (5.2), one can write

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = B_{\mathrm{rank}(\boldsymbol{\Sigma})}(0, \mathrm{rank}(\boldsymbol{\Sigma})) \cdot \sqrt{\boldsymbol{\Sigma}} + \boldsymbol{\mu} \subset B_{\mathrm{rank}(\boldsymbol{\Sigma})}(0, \mathrm{rank}(\boldsymbol{\Sigma})) \cdot \sqrt{\boldsymbol{\Sigma}} \pm \boldsymbol{\mu},$$

where $B_{\mathrm{rank}(\boldsymbol{\Sigma})}$ is defined in Eq. (1.1). And, with Equation Eq. (5.1), one can deduce $\mathrm{rank}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \mathrm{rank}(\boldsymbol{\Sigma}) + 1$, then:

$$B_{\mathrm{rank}(\boldsymbol{\Sigma})}(0, \mathrm{rank}(\boldsymbol{\Sigma})) \cdot \sqrt{\boldsymbol{\Sigma}} \pm \boldsymbol{\mu} \subset B_{\mathrm{rank}(\boldsymbol{\Sigma})+1}(0, \mathrm{rank}(\boldsymbol{\Sigma}) + 1) \cdot \begin{bmatrix} \sqrt{\boldsymbol{\Sigma}} \\ \boldsymbol{\mu} \end{bmatrix}.$$

We apply Definition 22 which confirms the inclusion of Eq. (5.4):

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \subset B_{\mathrm{rank}(\boldsymbol{\Sigma})+1}(0, \mathrm{rank}(\boldsymbol{\Sigma}) + 1) \cdot \begin{bmatrix} \sqrt{\boldsymbol{\Sigma}} \\ \boldsymbol{\mu} \end{bmatrix} = E(\mathbf{0}, \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}).$$

Thus, we can homogenize and transform the instance into a centered one with $\boldsymbol{\Sigma}' := \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}$.

Secondly, to get an isotropic distribution (i.e. with all its eigenvalues being 1), one can just multiply every element of the lattice with the pseudoinverse of $\sqrt{\boldsymbol{\Sigma}'}$. We get a new covariance matrix $\boldsymbol{\Sigma}'' = \sqrt{\boldsymbol{\Sigma}'}^{\sim} \cdot \boldsymbol{\Sigma}' \cdot \sqrt{\boldsymbol{\Sigma}'}^{\sim T} = \boldsymbol{\Pi}_{\boldsymbol{\Sigma}'} \cdot {\boldsymbol{\Pi}_{\boldsymbol{\Sigma}'}}^T$. And since $\boldsymbol{\Pi}_{\boldsymbol{\Sigma}'} = \boldsymbol{\Pi}_{\boldsymbol{\Sigma}'}^T$ and $\boldsymbol{\Pi}_{\boldsymbol{\Sigma}'}^2 = \boldsymbol{\Pi}_{\boldsymbol{\Sigma}'}$ (see Section 5.2.1), $\boldsymbol{\Sigma}'' = \boldsymbol{\Pi}_{\boldsymbol{\Sigma}'} = \boldsymbol{\Pi}_\Lambda$, the last equality coming from Eq. (5.1). In a nutshell, one must make by the two following changes:

$$\begin{aligned} \text{homogenize: } & (\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mapsto (\Lambda, \mathbf{0}, \boldsymbol{\Sigma}' := \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) \\ \text{isotropize: } & (\Lambda, \mathbf{0}, \boldsymbol{\Sigma}') \mapsto (\Lambda \cdot \mathbf{M}, \mathbf{0}, \boldsymbol{\Pi}_\Lambda) \end{aligned}$$

where $\mathbf{M} \coloneqq (\sqrt{\mathbf{\Sigma}'})^{\sim}$.

From the solution $\mathbf{x}$ to the $\mathsf{uSVP}_{\Lambda \cdot \mathbf{M}}$ problem, one can derive $\mathbf{x}' = \mathbf{x}\mathbf{M}^{\sim}$ the solution to the $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\mathbf{\Sigma}}$ problem. Note that, to solve a DBDD instance, one can ignore the isotropization and directly apply lattice reduction before the second step. This leads, however, to less efficient attacks.

Remark 22. *One should note that the first homogenization step "forgets" some information about the secret's distribution. This, however, is inherent to the conversion to a unique-SVP problem which is geometrically homogeneous and is already present in the original primal attack.*

5.3.4 Security estimates of uSVP: bikz versus bits

The attack on a uSVP instance consists of applying, the algorithm BKZ-β, presented in Section 1.2.2, on the uSVP lattice Λ for an appropriate block size parameter β. The cost of the attack grows with β, however, modeling this cost precisely is at the moment rather delicate, as the state of the art seems to still be in motion. Numerous NIST candidates choose to underestimate this cost, keeping a margin to accommodate future improvements, and there seems to be no clear consensus on which model to use (see [Alb+18] for a summary of existing cost models).

While this problem is orthogonal to our work, we still wish to be able to formulate quantitative security losses. We therefore express all concrete security estimates using the blocksize β as our measure of the level of security, and treat the latter as a measurement of the security level in a unit called the *bikz*. We thereby leave the question of the exact bikz-to-bit conversion estimate outside the scope of this chapter, and recall that those conversion formulae are not necessarily linear, and may have small dependency in other parameters. For the sake of concreteness, we note that certain choose, for example, to claim 128 bits of security for 380 bikz, and in this range, most models suggest a security increase of one bit every 2 to 4 bikz.

Remark 23. *We also clarify that the estimates given within this chapter only concern the pure lattice attack via the uSVP embedding discussed above. In particular, we note that some NIST candidates with ternary secrets [Lu+19] also consider the hybrid attack of [How07], which we ignore in this work. We nevertheless think that the compatibility with our framework is plausible, with some effort.*

Predicting β from a uSVP instance The state-of-the-art predictions for solving uSVP instances using BKZ were given in [Alk+16b; Alb+17]. Namely, for Λ a lattice of dimension $\dim(\Lambda)$, it is predicted that BKZ-β can solve a uSVP_{Λ} instance with secret $\mathbf{s}$ when

$$\sqrt{\beta/\dim(\Lambda)} \cdot \|\mathbf{s}\| \leq \delta_{\beta}^{2\beta-\dim(\Lambda)-1} \cdot \mathrm{Vol}(\Lambda)^{1/\dim(\Lambda)} \tag{5.5}$$

where δ_β is the so called root-Hermite factor of BKZ-β presented in Section 1.2.2. We recall that, for $\beta \geq 50$, the root-Hermite factor is predictable :

$$\delta_\beta = \left((\pi\beta)^{\frac{1}{\beta}} \cdot \frac{\beta}{2\pi e} \right)^{1/(2\beta-2)}. \tag{5.6}$$

Note that the uSVP instances we generate are isotropic and centered so that the secret has covariance $\mathbf{\Sigma} = \mathbf{I}$ (or $\mathbf{\Sigma} = \mathbf{\Pi}_\Lambda$ if Λ is not of full rank) and $\mu = \mathbf{0}$. Thus, on average, we

have $\|\mathbf{s}\|^2 = \text{rank}(\mathbf{\Sigma}) = \dim(\Lambda)$. Therefore, β can be estimated as the minimum integer that satisfies

$$\sqrt{\beta} \leq \delta_\beta^{2\beta-\dim(\Lambda)-1} \cdot \text{Vol}(\Lambda)^{1/\dim(\Lambda)}. \tag{5.7}$$

While β must be an integer as a BKZ parameter, we nevertheless provide a continuous value, for a finer comparison of the difficulty of an instance.

Remark 24. *To predict security, one does not need the basis of Λ, but only its dimension and its volume. Similarly, it is not necessary to explicitly compute the isotropization matrix $\mathbf{M}$ of Section 5.3.3, thanks to Lemma 22:* $\text{Vol}(\Lambda \cdot \mathbf{M}) = \text{rdet}(\mathbf{M})\,\text{Vol}(\Lambda) = \text{rdet}(\mathbf{\Sigma}')^{-1/2}\,\text{Vol}(\Lambda)$. *These two shortcuts will allow us to efficiently make predictions for cryptographically large instances, in our* lightweight *implementation of Section 5.5.*

Refined prediction for small blocksizes For experimental validation purposes of our work, we prefer to have accurate prediction even for small blocksizes; a regime where those predictions are not accurate with the current state of the art. Léo Ducas suggested a refined strategy using BKZ-simulation and a probabilistic model, we refer to the paper [Dac+20] for more details.

5.4 Including hints

In this Section, we define several categories of hints—**perfect hints**, **modular hints**, **approximate hints** (**conditioning** and ***a posteriori***), and **short vector hints**—and show that these types of hints can be integrated into a DBDD instance. Hints belonging to these categories typically have the form of a linear equation in $\mathbf{s}$ (and possibly additional variables). As emphasized in Section 5.1, these hints have lattice-friendly forms and their usefulness in realistic applications may not be obvious. We refer to Section 5.6 for detailed applications of these hints.

The technical challenge, therefore, is to characterize the effect of such hints on the DBDD instance—i.e. determine the resulting $(\Lambda', \boldsymbol{\mu}', \mathbf{\Sigma}')$ of the new DBDD instance, after the hint is incorporated.

Henceforth, let $\mathcal{I} = \text{DBDD}_{\Lambda,\boldsymbol{\mu},\mathbf{\Sigma}}$ be a fixed instance and let $\mathbf{s}$ be its secret solution. Each hint will introduce new constraints on $\mathbf{s}$ and will ultimately decrease the security level.

Non-Commutativity It should be noted that many types of hints commute: integrating them in any order will lead to the same DBDD instance. Potential exceptions are **numerical modular hints** (See later in Section 5.4.2) and ***a posteriori* approximate hints** (See later in Section 5.4.4): they do not always commute with the other types of hints, and do not always commute between themselves, unless the vectors $\mathbf{v}$'s of those hints are all orthogonal to each other. The reason is: in these cases, the distribution in the direction of $\mathbf{v}$ is redefined which erases the prior information.

5.4.1 Perfect Hints

Definition 26 (Perfect hint)**.** *A perfect hint on the secret $\mathbf{s}$ is the knowledge of $\mathbf{v} \in \mathbb{Z}^{d-1}$ and $l \in \mathbb{Z}$, such that*

$$\langle \mathbf{s},\ \mathbf{v} \rangle = l.$$

A perfect hint is quite strong in terms of additional knowledge. It allows decreasing the dimension of the lattice by one. One could expect such hints to arise from the following scenarios:

- The full leak without noise of an original coefficient, or even an unreduced intermediate register since most of the computations are linear. For the second case, one may note that optimized implementations of NTT typically attempt to delay the first reduction modulo q, so leaking a register on one of the first few levels of the NTT would indeed lead to such a hint.
- A noisy leakage of the same registers, but with still a rather high guessing confidence. In that case it may be worth making the guess while decreasing the success probability of the attack.[3] This could happen in a cold-boot attack scenario. This is also the case in the single trace attack on Frodo [Bos+18a] that we will study as one of our examples in Section 5.6.1.
- More surprisingly, certain schemes, including some NIST candidates offer such a hint 'by design'. Indeed, LAC and Round5 choose ternary secret vectors with a prescribed number of 1's and -1's, which directly induce one or two such perfect hints. This will be detailed in Section 5.6.3.

Integrating a perfect hint into a DBDD instance Let $\mathbf{v} \in \mathbb{Z}^{d-1}$ and $l \in \mathbb{Z}$ be such that $\langle \mathbf{s}, \mathbf{v} \rangle = l$. Note that the hint can also be written as

$$\langle \bar{\mathbf{s}},\ \bar{\mathbf{v}} \rangle = 0,$$

where $\bar{\mathbf{s}}$ is the extended LWE secret as defined in Eq. (5.3) and $\bar{\mathbf{v}} \coloneqq (\mathbf{v}\,;\, -l)$.

Remark 25. *Here we understand the interest of using Kannan's embedding* before *integrating hints rather than after: it allows to also homogenize the hint, and therefore to make Λ' a proper lattice rather than a lattice coset (i.e. a shifted lattice).*

Including this hint is done by modifying the $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ to $\mathsf{DBDD}_{\Lambda',\boldsymbol{\mu}',\boldsymbol{\Sigma}'}$, where:

$$\begin{aligned}
\Lambda' &= \Lambda \cap \left\{ \mathbf{x} \in \mathbb{Z}^d \mid \langle \mathbf{x}, \bar{\mathbf{v}} \rangle = 0 \right\} \\
\boldsymbol{\Sigma}' &= \boldsymbol{\Sigma} - \frac{(\bar{\mathbf{v}}\boldsymbol{\Sigma})^T \bar{\mathbf{v}}\boldsymbol{\Sigma}}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T} && (5.8) \\
\boldsymbol{\mu}' &= \boldsymbol{\mu} - \frac{\langle \bar{\mathbf{v}}, \boldsymbol{\mu} \rangle}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T} \bar{\mathbf{v}}\boldsymbol{\Sigma} && (5.9)
\end{aligned}$$

We now explain how to derive the new mean $\boldsymbol{\mu}'$ and the new covariance $\boldsymbol{\Sigma}'$. Let $\jmath$ be the random variable $\langle \bar{\mathbf{s}}, \bar{\mathbf{v}} \rangle$, where $\bar{\mathbf{s}}$ has mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Then $\boldsymbol{\mu}'$ is the mean of $\bar{\mathbf{s}}$ conditioned on $\jmath = 0$, and $\boldsymbol{\Sigma}'$ is the covariance of $\bar{\mathbf{s}}$ conditioned on $\jmath = 0$. Using Theorem 12, we obtain the corresponding conditional mean and covariance.

We note that lattice Λ' is an intersection of Λ and a hyperplane orthogonal to $\bar{\mathbf{v}}$. Given $\mathbf{B}$ as the basis of Λ, by Lemma 18, a basis of Λ' can be computed as follows:

1. Let $\mathbf{D}$ be dual basis of $\mathbf{B}$. Compute $\mathbf{D}_\perp \coloneqq \mathbf{D} \cdot \boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}}$.
2. Apply LLL algorithm on $\mathbf{D}_\perp$ to eliminate linear dependencies. Then delete the first row of $\mathbf{D}_\perp$ (which is $\mathbf{0}$ because with the hyperplane intersection, the dimension of the lattice is decremented).

[3] One may then re-amplify the success probability by retrying the attack making guesses at different locations

3. Output the dual of the resulting matrix.

While polynomial time, the above computation is quite heavy, especially as there is no convenient library offering a parallel version of LLL. Fortunately, for predicting attack costs, one only needs the dimension of the lattice Λ and its volume. These can easily be computed assuming $\bar{\mathbf{v}}$ is a primitive vector (See Definition 25) of the dual lattice: the dimension decreases by 1, and the volume increases by a factor $||\bar{\mathbf{v}}||$. This is stated and proved in Lemma 19. Intuitively, the primitivity condition is needed since then one can scale the leak to $\langle \mathbf{s}, f\mathbf{v} \rangle = fl$ for any non-zero factor $f \in \mathbb{R}$ and get an equivalent leak; however there is only one factor f that can ensure that $f\bar{\mathbf{v}} \in \Lambda^*$, and is primitive in it.

Remark 26. *Note that if $\bar{\mathbf{v}}$ is not in the span of Λ—as typically occurs if other non-orthogonal perfect hints have already been integrated—Lemma 19 should be applied to the orthogonal projection $\bar{\mathbf{v}}' = \bar{\mathbf{v}} \cdot \mathbf{\Pi}_\Lambda$ of $\bar{\mathbf{v}}$ onto Λ. Indeed, the perfect hint $\langle \bar{\mathbf{s}}, \bar{\mathbf{v}}' \rangle = 0$ replacing $\bar{\mathbf{v}}$ by $\bar{\mathbf{v}}'$ is equally valid.*

5.4.2 Modular Hints

Definition 27 (Modular hint)**.** *A modular hint on the secret $\mathbf{s}$ is the knowledge of $\mathbf{v} \in \mathbb{Z}^{d-1}$, $k \in \mathbb{Z}$ and $l \in \mathbb{Z}$, such that*

$$\langle \mathbf{s},\ \mathbf{v} \rangle = l \mod k.$$

We can expect such hints to arise from several scenarios:

- obtaining the value of an intermediate register during LWE decryption would likely correspond to giving such a modular equation modulo q. This is also the case if an NTT coefficient leaks in a Ring-LWE scheme. It can also occur "by design" if the LWE secret is chosen so that certain NTT coordinates are fixed to 0 modulo q, as is the case in some instances of Order LWE [Bol+19].

- obtaining the absolute value $a = |s|$ of a coefficient s implies $s = a \bmod 2a$, and such a hint could be obtained by a timing attack on an unprotected implementation of a table-based sampler, in the spirit of [Bru+16].

- obtaining the Hamming weight of the string $b_1 b_2 \ldots b'_1 b'_2 \ldots$ used to sample a centered binomial coefficient $s = \sum b_i - \sum b'_i$ (as done in NewHope and Kyber [Sch+19; Pöp+19]) reveals, in particular, $s \bmod 2$. Indeed, the latter string (or at least some parts of it) is more likely to be leaked than the Hamming weight of s.

Integrating a modular hint into a DBDD instance. Let $\mathbf{v} \in \mathbb{Z}^d$, $k \in \mathbb{Z}$ and $l \in \mathbb{Z}$ be such that $\langle \mathbf{s}, \mathbf{v} \rangle = l \mod k$. Note that the hint can also be written as

$$\langle \bar{\mathbf{s}},\ \bar{\mathbf{v}} \rangle = 0 \mod k \tag{5.10}$$

where $\bar{\mathbf{s}}$ is the extended LWE secret as defined in Eq. (5.3) and $\bar{\mathbf{v}} := (\mathbf{v}\,;\, -l)$. We refer to Remark 25 for the legitimacy of such dimension increase.

Intuitively, such a hint should only sparsify the lattice, and leave the average and the variance unchanged. This is not entirely true, this is only (approximately) true when the variance is sufficiently large in the direction of $\mathbf{v}$ to ensure smoothness, i.e. when

$k^2 \ll \mathbf{v}\mathbf{\Sigma}\mathbf{v}^T$; one can refer to [MR07, Lemma 3.3 and Lemma 4.2] for the quality of that approximation. In this smooth case, we therefore have:

$$\Lambda' = \Lambda \cap \left\{\mathbf{x} \in \mathbb{Z}^d \mid \langle \mathbf{x}, \mathbf{v}\rangle = 0 \mod k\right\} \tag{5.11}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} \tag{5.12}$$

$$\mathbf{\Sigma}' = \mathbf{\Sigma} \tag{5.13}$$

On the other hand, if $k^2 \gg \mathbf{v}\mathbf{\Sigma}\mathbf{v}^T$, then the residual distribution will be highly concentrated on a single value, and one should therefore instead use a perfect $\langle \mathbf{s},\ \mathbf{v}\rangle = l + ik$ for some i.

Numerical solution. One can wonder how to include the hint if the smoothness is not ensured and if it is not a perfect hint. One can always resort to a numerical computation of the average μ_c and the variance σ_c^2 of the one-dimensional centered discrete Gaussian of variance $\sigma^2 = \mathbf{v}\mathbf{\Sigma}\mathbf{v}^T$ over the coset $l + k\mathbb{Z}$. Thus, the lattice should then be sparsified as in the smooth case with Equation Eq. (5.11) and the following corrections must be applied:

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \frac{\mu_c - \langle \bar{\mathbf{v}}, \boldsymbol{\mu}\rangle}{\bar{\mathbf{v}}\mathbf{\Sigma}\bar{\mathbf{v}}^T}\bar{\mathbf{v}}\mathbf{\Sigma} \tag{5.14}$$

$$\mathbf{\Sigma}' = \mathbf{\Sigma} + \left(\frac{\sigma_c^2}{(\bar{\mathbf{v}}\mathbf{\Sigma}\bar{\mathbf{v}}^T)^2} - \frac{1}{\bar{\mathbf{v}}\mathbf{\Sigma}\bar{\mathbf{v}}^T}\right)(\bar{\mathbf{v}}\mathbf{\Sigma})^T(\bar{\mathbf{v}}\mathbf{\Sigma}) \tag{5.15}$$

$$\tag{5.16}$$

Intuitively, these formulae completely erase prior information on $\langle \mathbf{s}, \bar{\mathbf{v}}\rangle$, before it is replaced by the new average and variance in the adequate direction. Both can be derived using Theorem 12.

As for perfect hints, the computation of Λ' can be done by working on the dual lattice. More specifically:

1. Let $\mathbf{D}$ be dual basis of $\mathbf{B}$.
2. Redefine $\bar{\mathbf{v}} \coloneqq \bar{\mathbf{v}} \cdot \mathbf{\Pi}_\Lambda$, noting that this does not affect the validity of the hint.
3. Append $\bar{\mathbf{v}}/k$ to $\mathbf{D}$ and obtain $\mathbf{D}'$
4. Apply LLL algorithm on $\mathbf{D}'$ to eliminate linear dependencies. Then delete the first row of $\mathbf{D}'$ (which is $\mathbf{0}$ since we introduced a linear dependency).
5. Output the dual of the resulting matrix.

Also, as for perfect hints, the parameters of the new lattice Λ' can be predicted: the dimension is unchanged, and the volume increases by a factor k under a primitivity condition, which is proved by Lemma 20.

5.4.3 Approximate Hints (conditioning)

Definition 28 (Approximate hint)**.** *An approximate hint on secret* $\mathbf{s}$ *is the knowledge of* $\mathbf{v} \in \mathbb{Z}^{d-1}$ *and* $l \in \mathbb{Z}$*, such that*

$$\langle \mathbf{s},\ \mathbf{v}\rangle + e = l,$$

where e *models noise following a centered normal distribution of standard deviation* σ_e^2*, independent of* $\mathbf{s}$*.*

One can expect such hints from:

- any noisy side channel information about a secret coefficient. This is the case of our study in Section 5.6.1.
- decryption failures. In Section 5.6.2, we show how this type of hint can represent the information gained by a decryption failure.

To include this knowledge in the DBDD instance, we must combine this knowledge with the prior knowledge on the solution $\mathbf{s}$ of the instance.

Integrating an approximate hint into a DBDD instance Let $\mathbf{v} \in \mathbb{Z}^{d-1}$ and $l \in \mathbb{Z}$ be such that $\langle \mathbf{s}, \mathbf{v} \rangle \approx l$. Note that the hint can also be written as

$$\langle \bar{\mathbf{s}},\ \bar{\mathbf{v}} \rangle + e = 0 \tag{5.17}$$

where $\bar{\mathbf{s}}$ is the extended LWE secret as defined in Eq. (5.3), $\bar{\mathbf{v}} \coloneqq (\mathbf{v}\,;\, -l)$, and e has $N_1(0, \sigma_e^2)$ distribution. The unique shortest non-zero solution of $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$, is also the unique solution of the instance $\mathsf{DBDD}_{\Lambda',\boldsymbol{\mu}',\boldsymbol{\Sigma}'}$ where

$$\Lambda' = \Lambda \tag{5.18}$$

$$\boldsymbol{\Sigma}' = \boldsymbol{\Sigma} - \frac{(\bar{\mathbf{v}}\boldsymbol{\Sigma})^T \bar{\mathbf{v}}\boldsymbol{\Sigma}}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T + \sigma_e^2} \tag{5.19}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} - \frac{\langle \bar{\mathbf{v}}, \boldsymbol{\mu} \rangle}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T + \sigma_e^2}\bar{\mathbf{v}}\boldsymbol{\Sigma} \tag{5.20}$$

We note that Eq. (5.18) comes from

$$\Lambda' \coloneqq \Lambda \cap \left\{ \mathbf{x} \in \mathbb{Z}^d \mid \langle \mathbf{x}, \bar{\mathbf{v}} \rangle + e = 0,\ \text{for all possible } e \sim N_1(0, \sigma_e^2) \right\} = \Lambda.$$

The new covariance and mean follow from Theorem 12.

Consistency with Perfect Hint Note that if $\sigma_e = 0$, we fall back to a perfect hint $\langle \mathbf{s}, \mathbf{v} \rangle = l$. The above computation of $\boldsymbol{\Sigma}'$ Eq. (5.19) (resp. $\boldsymbol{\mu}'$ Eq. (5.20)) is indeed equivalent to Eq. (5.8) (resp. Eq. (5.9)) from Section 5.4.1. Note however, in our implementation, that to avoid singularities, we require the span of $\mathrm{Span}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T\boldsymbol{\mu}) = \mathrm{Span}(\Lambda)$ (See the requirement in Equation Eq. (5.1)): If $\sigma_e = 0$, one *must* instead use a Perfect hint.

Multi-dimensional approximate hints The formulae of [Liu19] are even more general, and one could consider a multidimensional hint of the form $\mathbf{sV} + \mathbf{e} = \mathbf{l}$, where $\mathbf{V} \in \mathbb{R}^{n \times k}$ and $\mathbf{e}$ a Gaussian noise of any covariance $\boldsymbol{\Sigma}_\mathbf{e}$. However, those general formulae require explicit matrix inversion which becomes impractical in large dimension. We therefore only implemented full-dimensional ($k = n$) hint integration in the *super-lightweight* version of our tool, which assumes all covariance matrices to be diagonal. These will be used for hints obtained from decryption failures in Section 5.6.2.

5.4.4 Approximate Hint (*a posteriori*)

In certain scenarios, one may more naturally obtain directly the a posteriori distribution of $\langle \mathbf{s}, \mathbf{v} \rangle$, rather than a hint $\langle \mathbf{s}, \mathbf{v} \rangle + e = l$ for some error e independent of $\mathbf{s}$. Such a

scenario is typical in template attacks, as we exemplify via the single trace attack on Frodo from [Bos+18a], which we study in Section 5.6.1.

Given the a posteriori distribution of $\langle \bar{\mathbf{s}}, \bar{\mathbf{v}} \rangle$, one can derive its mean μ_{ap} and variance σ^2_{ap} and apply the corrections to compute the new mean and covariance exactly as in Eqs. (5.14) and (5.15).

$$\Lambda' = \Lambda \tag{5.21}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \frac{\mu_{\text{ap}} - \langle \bar{\mathbf{v}}, \boldsymbol{\mu} \rangle}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T} \bar{\mathbf{v}}\boldsymbol{\Sigma} \tag{5.22}$$

$$\boldsymbol{\Sigma}' = \boldsymbol{\Sigma} + \left(\frac{\sigma^2_{\text{ap}}}{(\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T)^2} - \frac{1}{\bar{\mathbf{v}}\boldsymbol{\Sigma}\bar{\mathbf{v}}^T} \right) (\bar{\mathbf{v}}\boldsymbol{\Sigma})^T (\bar{\mathbf{v}}\boldsymbol{\Sigma}) \tag{5.23}$$

5.4.5 Short vector hints

Definition 29 (Short vector hint)**.** *A short vector hint on the lattice Λ is the knowledge of a short vector $\mathbf{v}$ such that*

$$\mathbf{v} \in \Lambda.$$

Note that such hints are not related to the secret, and are not expected to be obtained by side-channel information, but rather by the very design of the scheme. In particular, the lattice Λ underlying LWE instance modulo q contains the so-called q-vectors, i.e. the vectors $(q, 0, 0, \ldots, 0)$ and its permutations. These vectors are in fact implicitly exploited in the literature on the cryptanalysis of LWE since at least [LP11]. Indeed, in some regimes, the best attacks are obtained by 'forgetting' certain LWE equations, which can be geometrically interpreted as a projection orthogonally to that vector. Note that, among all hints, the short vector hints should be the last to be integrated. In our context, we need to generalize this idea beyond q-vector because the q-vectors may simply disappear after the integration of a perfect or modular hint. For example, after the integration of a perfect hint $\langle \mathbf{s}, (1, 1, \ldots, 1) \rangle = 0$, all the q-vectors are no longer in the lattice, but $(q, -q, 0, \ldots, 0)$ still is, and so are all its permutations.

Resolving the DBDD problem resulting from this projection will not directly lead to the original secret, as projection is not injective. However, as long as we keep $n + 1$ dimensions out of the $n + m + 1$ dimensions of the original LWE instance, we can still efficiently reconstruct the full LWE secret by solving a linear system over the rationals.

Integrating a short vector hint into a DBDD instance It is the case when the secret vector is short enough to be a solution after applying projection $\boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}}$ on $\mathsf{DBDD}_{\Lambda,\boldsymbol{\Sigma},\boldsymbol{\mu}}$.

$$\Lambda' = \Lambda \cdot \boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}} \tag{5.24}$$

$$\boldsymbol{\Sigma}' = (\boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}})^T \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}} \tag{5.25}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} \cdot \boldsymbol{\Pi}^{\perp}_{\bar{\mathbf{v}}} \tag{5.26}$$

To compute a basis of Λ', one can simply apply the projection to all the vectors of its current basis, and then eliminate linear dependencies in the resulting basis using LLL.

Remark 27. *Once a short vector hint $\mathbf{v} \in \Lambda$ has been integrated, Λ has been transformed into Λ'. And, if one has to perform another short vector hint integration $\mathbf{v}_1 \in \Lambda$, $\mathbf{v}_1$ should be projected onto Λ' with $\mathbf{v} \cdot \boldsymbol{\Pi}_{\Lambda'} \in \Lambda'$. In our implementation, however, this has been taken into account and one can simply apply the same transformation as above, replacing a single vector $\mathbf{v}$ by a matrix $\mathbf{V}$.*

The dimension of the lattice decreases by one (or by k, if one directly integrates a matrix of k vectors) and the volume of the lattice also decreases according to Lemma 21.

Worthiness and choice of short vector hints Integrating such a hint induces a trade-off between the dimension and the volume, and therefore it is not always advantageous to integrate. We refer to [Dac+20] for a discussion on the worthiness of the short vector hints. In the typical studied cases, the short vector hints are either the q-vectors or some simple transformation of the q-vectors (See Section 5.6.3 for example).

5.5 Implementation

5.5.1 Our sage implementation

We propose three implementations of our framework, all following the same python/sage 9.0 API.[4] More specifically, the API and some common functions are defined in a sage file denoted DBDD_generic.sage, as a class `DBDD_Generic`. Three derived classes are then given:

1. The class `DBDD` (provided in DBDD.sage) is the *full-fledged* implementation: i.e. it fully maintains all information about a DBDD instance as one integrates hints: the lattice Λ, the covariance matrix $\mathbf{\Sigma}$ and the average μ. While polynomial time, maintaining the lattice information can be quite slow, especially since consecutive intersections with hyperplanes can lead to manipulations on rationals with large denominators. It also allows to finalize the attack, running the homogenization, isotropization and lattice reduction, based on the fplll library available through sage.

   We note that if one were to repeatedly use perfect or modular hints, a lot of effort would be spent on uselessly alternating between the primal and the dual lattice. Instead, we implement a caching mechanism for the primal and dual basis, and only update them when necessary.

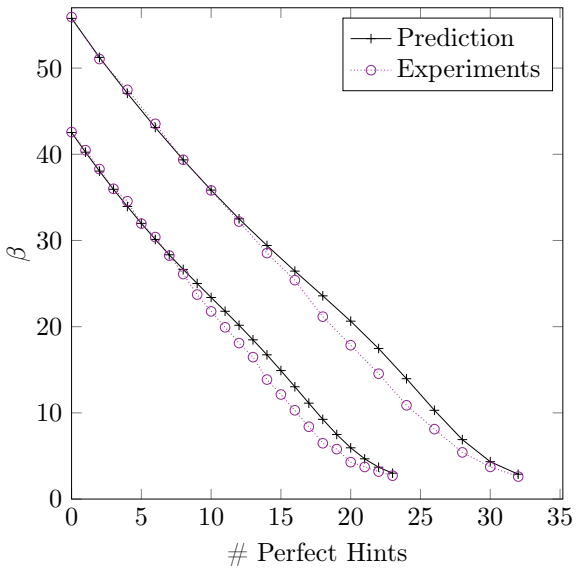
2. The class `DBDD_predict` (provided in DBDD_predict.sage) is the *lightweight* implementation: it only fully maintains the covariance information, and the parameters of the lattice (dimension, volume). It must therefore work under assumptions about the primitivity of the vector $\mathbf{v}$; in particular, it cannot detect hints that are redundant. If one must resort to this faster variant on large instances, it is advised to consider potential (even partial) redundancy between the given hints, and to run a comparison with the previous on small instances with similarly generated hints.

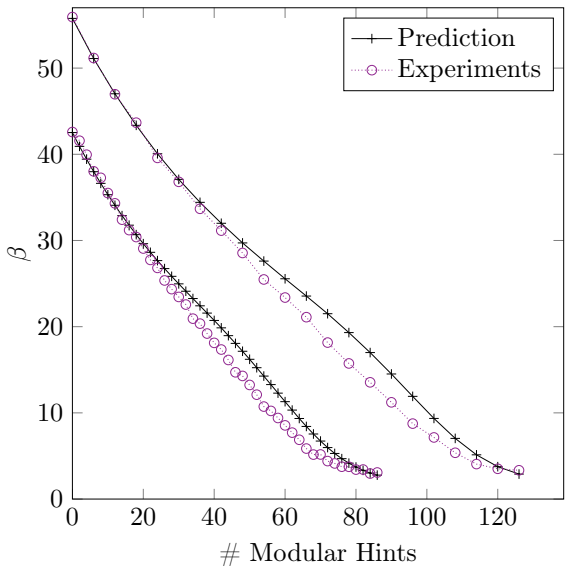
3. The class `DBDD_predict_diag` (provided in DBDD_predict_diag.sage) is the *super-lightweight* implementation. It maintains the same information as the above, but requires the covariance matrix to remain diagonal at all times. In particular, one can only integrate hints for which the directional vector $\mathbf{v}$ is colinear with a canonical vector.

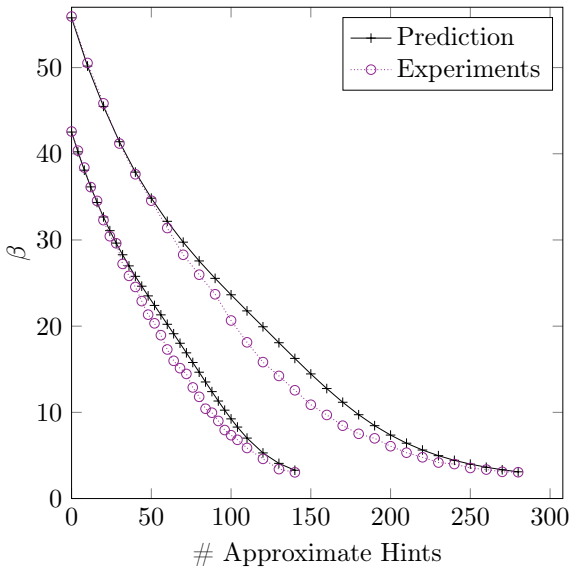
5.5.2 Tests and validation

In our paper [Dac+20], we present a demonstration of our tool with some extracts of Sage 9.0 code. We implement two tests to verify the correctness of our scripts, and more generally the validity of our predictions.

[4] While we would have preferred a full python implementation, we are making a heavy use of linear algebra over the rationals for which we could find no convenient python library.







LWE Parameters		
$n = m = 70$,	$q = 3301$,	$\sigma = 20$
$n = m = 80$,	$q = 3301$,	$\sigma = 20$

Hint Type	Parameter
Perfect hints	–
Modular hints	mod 11
Approximate hints	$\sigma_\epsilon^2 = 3$

The hint vectors $\mathbf{v}$ were chosen as random ternary vectors of weight 5.

Figure 5.5: *Experimental verification of the security decay predictions for each type of hint. Each data point was averaged over 256 samples.*

Consistency checks. Our first test simply verifies that all three classes always agree perfectly. More specifically, we run all three versions on a given instance, integrating the same random hint in all of them, and compare their hardness prediction. We first test using the full-fledged version that the primitivity condition does hold, and discard the hint if not, as we know that predictions cannot be correct on such hints. This verification passes.

Prediction verifications. We now verify experimentally the prediction made by our tool for various types of hints, by comparing those predictions to actual attack experiments. This is done for a given set of LWE parameters, and increasing the number of hints. The details of the experiments and the results are given in Fig. 5.5.

While our predictions seem overall accurate, we still note a minor discrepancy of up to 2 or 3 bikz in the low blocksize regime. This exceeds the error made by prediction on the attack without any hint, which was below 1 bikz, even in the same low blocksize regime.

We suspect that this discrepancy is due to residual q-vectors, or small combinations of them, that are hard to predict for randomly generated hints but would still benefit by lattice reduction. We tested that hypothesis by running similar experiments, but leaving certain coordinates untouched by hints, so to still explicitly know some q-vectors for short-vector hint integration, if they are "worthy". This did not to improve the accuracy of our prediction. We are at the moment unable to explain it. We nevertheless find our predictions satisfactory, considering that even without considering hints, previous predictions [Alb+17] were much less accurate.

5.6 Applications

The purpose of this section is to exemplify our tool on a several known side-channel, decryption-failures, or classical attacks. We chose several simple scenarios already existing in the literature to show how our framework can be used. Even though the results in themselves are not claimed to beat the best known attacks, we believe that it opens the path for more applications and more elaborate hybrid lattice attacks.

5.6.1 Hints from side channels: application to [Bos+18a]

In [Bos+18a], W. Bos et al. study the feasibility of a single-trace power analysis of the Frodo Key Encapsulation Mechanism [Nae+19]. We refer to Section 1.1.3 for more details on this single-trace attack model. Specifically, in a first approach, they analyze the possibility of a divide-and-conquer attack targeting a multiplication in the key generation. This attack was claimed unsuccessful in [Bos+18a] because the brute force phase after recovering a candidate for the private key was too expensive. Along with this unsuccessful result, a successful powerful extend-and-prune attack is provided in [Bos+18a].

> Keeping the purpose of this section in mind, improving the former unsuccessful divide-and-conquer attack of [Bos+18a] is a perfect application for understanding the impact of our framework. However, note that we do not claim to have found *the best* single-trace attack on Frodo as more powerful dedicated attacks, like in [Bos+18a], exist.

Frodo KEM. FrodoKEM is based on small-secret-LWE (Hard Problem 10), we outline here some details necessary to understand the attack. Note that we use different letter notations from [Nae+19] for consistency with this chapter. For parameters n and q, the private key is $(\mathbf{z} \in \mathbb{Z}_q^n, \mathbf{e} \in \mathbb{Z}_q^n)$ where the coefficients of $\mathbf{z}$ and $\mathbf{e}$, denoted $\mathbf{z}_i$ and $\mathbf{e}_i$, can take several values in a small set that we denote L. The public key is $\big(\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{b} = \mathbf{z}\mathbf{A} + \mathbf{e}\big)$. The goal of the attack is to recover $\mathbf{z}$ by making measurements during the multiplication between $\mathbf{z}$ and $\mathbf{A}$ when computing $\mathbf{b}$ in the key generation. Note that there is no multiplication involving $\mathbf{e}$ and thus we do not target it in this attack. Six sets of parameters are considered: CCS1, CCS2, CCS3 and CCS4 introduced in [Bos+16] and NIST1 and NIST2 introduced in [Nae+17]. For example, with NIST1 parameters,

$$n = 640,\ q = 2^{15} \text{ and } L = \{-11, \cdots, 11\}.$$

Side-channel simulation. The divide-and-conquer attack provided by [Bos+18a] simulates side-channel information using ELMO, a power simulator for a Cortex M0 [MOW17].

This tool outputs simulated power traces using an elaborate leakage model with Gaussian noise. Thus, it is parameterized by the standard deviation of the side-channel noise. For proofs of concept, the authors of [MOW17] suggest to choose the standard deviation of the simulated noise as

$$\sigma_{\text{SimNoise}} \coloneqq 0.0045,$$

for a realistic leakage modelization. This standard deviation was also the one chosen in [Bos+18a, Fig. 2b] and W. Bos et al. implemented a Matlab script, that calls ELMO, to simulate the side-channel information applied on Frodo. This precise side-channel simulator was provided to us by the authors of [Bos+18a] and we were able to re-generate all their data with Matlab still using $\sigma_{\text{SimNoise}} = 0.0045$.

Template attack. The divide-and-conquer side-channel attack proposed by W. Bos et al. belongs in the template attack family. Template attacks were introduced in [CRR03]. In a nutshell, these attacks include a profiling phase and an online phase. Let us detail the template divide-and-conquer attack for Frodo implemented in [Bos+18a].

1. The profiling phase consists in using a copy of the device and recording a large number of traces using many different known secret values. From these measures, the attacker can derive the multidimensional distribution of several interest points when the traces share a same secret coefficient. More precisely, in the case of FrodoKEM, for a given index $i \in [0, n-1]$, the interest points will be the instants in the trace when $\mathbf{z}_i$ is multiplied by the coefficients of $\mathbf{A}$ (n interest points in total). Let us define

$$\mathbf{c}_i \coloneqq (T[t_{i,0}], \ldots, T[t_{i,n-1}]) \quad \mathbf{c} \in \mathbb{R}^n, \tag{5.27}$$

   where T denotes the trace measurement and $(t_{i,k})$ denote the instants of the mutiplication of $\mathbf{z}_i$ with the coefficients $\mathbf{A}_{i,k}$ for $(i,k) \in [0, n-1]$. The random variable vector associated to $\mathbf{c}_i$ is denoted $\mathfrak{C}_i$. For each $i \in [0, n-1]$ and $x \in L$, the goal of the profiling phase is to learn the center of the probability distribution

$$\mathcal{D}_{i,x}(\mathbf{c}) \coloneqq P\left[\mathfrak{C}_i = \mathbf{c} \mid \mathbf{z}_i = x\right].$$

   By hypothesis for template attacks (see [CRR03, Section 2.1]), $\mathcal{D}_{i,x}$ is assumed to follow a multidimenstional normal distribution of standard deviation σ_{SimNoise}. Thus, the attacker recovers the center of $\mathcal{D}_{i,x}$ for each $i \in [0, n-1]$ and $x \in L$ by averaging all the measured $\mathbf{c}_i$ that validate $\mathbf{z}_i = x$. The center of $\mathcal{D}_{i,x}$ is denoted $\mathbf{t}_{i,x}$ and we call it a *template*. Before going further, [Bos+18a] actually make the following independence assumption.

   Assumption 5. $\mathbf{t}_{i,x}$ only depends on x and is independent from the index i. Thus, $\mathbf{t}_{i,x} = \mathbf{t}_x$.

   Essentially, this common assumption considers that the index $i \in [0, n-1]$ of the target coefficient do not influence the leakage. Consequently, the attacker only has to derive $\mathbf{t}_{0,x}$ for example.

2. In a second step, the attacker knows the templates $\mathbf{t}_x$ for all $x \in L$. She also knows the interest points $t_{i,k}$ as defined above in Eq. (5.27). She will construct a candidate $\tilde{\mathbf{z}}$ for the secret $\mathbf{z}$ by recovering the coefficients one by one. For each unknown secret coefficient $\mathbf{z}_i$, she makes the measurement $\mathbf{c}_i$ as defined in Eq. (5.27). Using this

Table 5.1: *Examples of scores associated to the secret values $\mathbf{s}_i \in \{0, \pm 1\}$, after the side-channel analysis of [Bos+18a] for NIST1 parameters. The best score in each score table is highlighted. This best guess is correct for the first 3 score table, but incorrect for the last one.*

$\mathbf{z}_i$	S											
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
0	-4098	-3918	-4344	-2580	-3212	-3108	-3758	-3155	-3583	-3498	-3900	**-340**
1	-3273	-3114	-3491	-1951	-2495	-2405	-2972	-2445	-2819	-2744	-3098	-365
−1	-341	-335	-352	-465	-358	-369	-329	-362	-331	-334	**-328**	-3712
−1	-306	-298	-319	-414	-314	-323	**-290**	-317	-291	-293	-291	-3608

	...	1	2	3	4	5	6	7	8	9	10	11
0	...	-380	-367	-452	-818	-975	-933	-1084	-368	-459	-453	-592
1	...	**-325**	-328	-338	-546	-657	-627	-737	-333	-344	-342	-407
−1	...	-3079	-3195	-2656	-1696	-1461	-1521	-1329	-3231	-2648	-2685	-2201
−1	...	-2982	-3097	-2564	-1617	-1385	-1444	-1256	-3132	-2556	-2593	-2115

measurement, she can derive an a posteriori probability distribution: with her fixed $i \in [0, n-1]$ and measured $\mathbf{c}_i \in \mathbb{R}$, she computes for all $x \in L$,

$$P\left[\mathbf{z}_i = x \mid \mathfrak{C}_i = \mathbf{c}_i\right] = \frac{P\left[\mathbf{z}_i = x\right]}{P\left[\mathfrak{C}_i = \mathbf{c}_i\right]} \cdot P\left[\mathfrak{C}_i = \mathbf{c}_i \mid \mathbf{z}_i = x\right] \tag{5.28}$$

$$\propto P\left[\mathbf{z}_i = x\right] \cdot \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{t}_x\|_2^2}{2\sigma_{\text{SimNoise}}^2}\right) \tag{5.29}$$

In [Bos+18a], a score table, denoted $(S_i[x])_{x \in L}$ is derived from the a posteriori distribution as follows,

$$S_i[x] \coloneqq \ln\left(P\left[\mathbf{z}_i = x \mid \mathfrak{C}_i = \mathbf{c}_i\right]\right) \tag{5.30}$$

$$= \ln\left(P\left[\mathbf{z}_i = x\right]\right) - \frac{\|\mathbf{c}_i - \mathbf{t}_x\|_2^2}{2\sigma_{\text{SimNoise}}^2}. \tag{5.31}$$

Finally, the output candidate for $\mathbf{z}_i$ is $\tilde{\mathbf{z}}_i \coloneqq \text{argmax}_{x \in L}(S_i[x])$.

One can use the presented attack in "black-box" to generate the score tables using the script from [Bos+18a]. As an example, using the NIST1 parameters, show several measured scores $(S[-11], \cdots, S[11])$ corresponding to several secret coefficients in Table 5.1. The first line corresponds to a secret equal to 0, the second line to 1 and the third and fourth line to -1. The last line is an example of failed guessing because we see that the outputted candidate is not -1. We remark that the values having the opposite sign are assigned a very low score, we conjecture that it is because the sign is filling the register and then the Hamming weight of the register will be very far from the correct one.

With this template attack, one can recover $\tilde{\mathbf{z}} \approx \mathbf{z}$. However, W. Bos et al. could not conclude the attack with a key recovery even though much information leaked about the secret. Frustratingly, a brute force phase to derive $\mathbf{z}$ from $\tilde{\mathbf{z}}$ cannot lead to any security threat as stated in [Bos+18a, Section 3]. They actually pointed out an interesting open question: "possibly [...] novel lattice reduction algorithms [can] take into account side-channel information". The following solves this open question by combining the knowledge obtained in the divide-and-conquer template attack of [Bos+18a] with our framework.

Table 5.2: *Probability distributions derived from Table 5.1, along with variances and centers.*

$\mathbf{z}_i$	A posteriori distribution											
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0
−1	0	0.001	0	0	0	0	0.258	0	0.035	0.002	0.704	0
−1	0	0	0	0	0	0	0.560	0	0.206	0.028	0.206	0

	...	1	2	3	4	5	6	7	8	9	10	11	center	variance
0	...	0	0	0	0	0	0	0	0	0	0	0	0	0
1	...	0.95	0.4	0	0	0	0	0	0.1	0	0	0	1.05	0.06
−1	...	0	0	0	0	0	0	0	0	0	0	0	-2.11	3.11
−1	...	0	0	0	0	0	0	0	0	0	0	0	-3.68	2.63

Table 5.3: *Cost of the attacks without/with hints without/with guesses.*

		NIST1	NIST2	CCS1	CCS2	CCS3	CCS4
Attack without hints	(bikz)	487	708	239	448	492	584
Attack with hints	(bikz)	**330**	**423**	**128**	**123**	**219**	**230**
Attack with hints & guesses	(bikz)	**292**	**298**	**70**	**29**	**124**	**129**
Number of guesses g		100	250	200	300	250	250
Success probability		0.86	0.64	0.87	0.77	0.81	0.84

Our refinement. We first instantiate a DBDD instance with a chosen set of parameters. Then we assume that, for each secret coefficient $\mathbf{z}_i$, we are given the associated score table S_i thanks to the presented template attack. We simply go back to the a posteriori distribution in Eq. (5.29) by applying the exp() and renormalizing the score table. As an example, we show the probability distributions derived from Table 5.1, along with their variances and centers in Table 5.2.

Finally, we use our framework to introduce n a posteriori *approximate hints* to our DBDD instance with the derived centers and variances for each score table. When the variance is exactly 0, we integrate perfect hints instead.

Results. One can note that the obtained security fluctuates a bit from instance to instance, as it depends on the strength of the hints, which themselves depend on the randomness of the scheme. In the first two lines of Table 5.3, we show the new security with the inclusion of the approximate hints averaged on 50 tests per set of parameters.

Guessing. To improve the attack further, one can note from Table 5.2 that certain key values have a very high probability of being correct, and assuming it is, one can replace an approximate hint with a perfect one. For example, considering the second line of Table 5.2, the secret has a probability of 0.95 to be 1 and thus guessing it trades a perfect hint for a decrease of the success probability of the attack by 5%. This hybrid attack exploiting hints, guesses and lattice reduction, works as follows. Let g be a parameter.

1. Include all the approximate and perfect hints given by the score tables,
2. Order the coefficients of the secret $\mathbf{z}_i$ according to the maximum value of their a posteriori distribution table,

3. Include perfect hints for the g first coefficients and then solve and check the solution.

Increasing the number of guesses g leads to a trade-off between the cost of the attack and its success probability. We have chosen here a success probability larger than 0.6, while reducing the attack cost by 38 to 145 bikz depending on the parameter set. Given that 1 bit of security corresponds roughly to 3 or 4 bikz, this is undoubtedly advantageous.

The results presented above are very recent (lastly improved on May the 20th 2020). We remark that, with these results, the attacks with guesses on the parameters CCS1 and CCS2 seem doable in practice while it was not the case with our original results. However, some improvements of the implementation remain to be done in order to actually mount the attack. The full-fledged implementation cannot handle the large matrices of the original DBDD instance. We require another class of implementation which fully maintains all information about the instance, like the `DBDD` class, and assumes that the covariance matrix $\mathbf{\Sigma}$ is diagonal to simplify the computations, like the `DBDD_predict_diag` class.

Perspective 8 (Second-guessing.)**.** *It should be noted that, given a single trace, one cannot naively retry the attack to boost its success probability. Indeed, the "second-best" guess may already have a much lower success probability than the first, and an interesting perspective boils down to setting up a hybrid attack mixing lattice reduction within our framework, and key-ranking.*

5.6.2 Hints from decryption failures

Another kind of hint our framework can model are hints provided by decryption failures.

Application to [DAn+19a]

In addition to the immediate application of our framework on Section 4.3's decryption failure attack (see Section 4.3.3), one can also apply our framework to the anterior work of D'Anvers et al. [DVV18], as originally presented in our paper [Dac+20]. In the latter, to compare the predictions given by our tool to the ones from [DVV18], we chose FRODOKEM-976 parameters, for which we were able to reproduce the data[5]. We note that our framework produces essentially similar predictions, as shown in Fig. 5.6, and refer to our paper for more details on the technique.

Application to the results of Section 4.2.3 ([Bau+19])

As a contribution of this thesis, we apply our framework to the work of Section 4.2.3 for an alternative way to recover the private key. In Section 4.2.3, let k be fixed in $[0, 255]$, the attacker wants to recover the secret $\mathbf{S}[k] \in [-8, 8]$ and she has access to the sign of a function f_{ℓ_1,ℓ_2,ℓ_3} that can be written as

$$f_{\ell_1,\ell_2,\ell_3}(\ell_0) = \underbrace{V_{\ell_1,\ell_2,\ell_3}}_{\text{constant}} + \left| \ell_0 - \frac{\overbrace{\mathbf{S}[k]}^{\in[-8,8]}}{2} + \frac{1 + Sign(\ell_0)}{2s} \right| .$$

[5] https://github.com/KULeuven-COSIC/PQCRYPTO-decryption-failures/

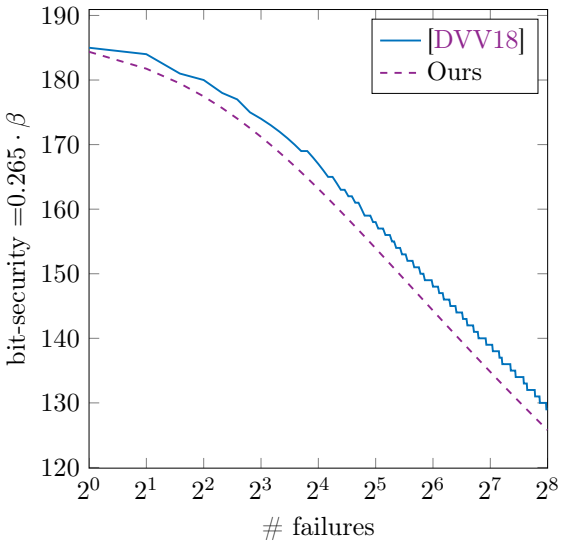


Figure 5.6: *Security decrease as a function of the number of failures in FRODOKEM-976.*

As shown in [Bau+19], depending on the value of V_{ℓ_1,ℓ_2,ℓ_3} that can be negative, two changes of sign can happen when iterating $\ell_0 = \{-4, \dots 3\}$. If two changes of sign do not happen, we redraw the values of (ℓ_1, ℓ_2, ℓ_3) until two changes of sign occur (we refer to [Bau+19, Section 4.2] for proving that it terminates). Therefore, let us assume that there are two changes of sign, which is obtained after ≈ 18 decryption oracle queries on average. So, this strategy takes around $18 \times 1024 \approx 18,500$ oracle queries, referring to [Bau+19] for more details about the queries estimation.

Similarly to Section 5.6.1, we perform a learning phase to derive an a posteriori distribution knowning the position of the changes of sign of f_{ℓ_1,ℓ_2,ℓ_3}. Before the attack, with a functional analysis, one can learn the possible position of the changes of sign with the knowledge of the value of $\mathbf{S}[k]$ with the study of the variations of f_{ℓ_1,ℓ_2,ℓ_3}. For example, if $\mathbf{S}[k] = -5$, the function can be written as follows

$$f_{\ell_1,\ell_2,\ell_3}(\ell_0) = V_{\ell_1,\ell_2,\ell_3} + F[\ell_0 + 4]$$

with $F = [1.5, 0.5, 0.5, 1.5, 2.50065, 3.50065, 4.50065, 5.50065]$. Thus, if there are two changes of sign when iterating f_{ℓ_1,ℓ_2,ℓ_3} with $\ell_0 = \{-4, \dots 3\}$, the only possible issue is that $f_{\ell_1,\ell_2,\ell_3}(\ell_0) < 0$ for $\ell_0 \in \{-3, -2\}$. From this learning phase, one can derive the aposteriori knowledge of the secret coefficient when knowing the position of the change of sign. More precisely, we obtain an aposteriori knowledge presented in Table 5.4.

Table 5.4: *Aposteriori knowledge after Section 4.2.3 attack*

Values ℓ_0 that verify $f_{\ell_1,\ell_2,\ell_3}(\ell_0) < 0$	$\{0,1,2\}$	$\{-3,-2\}$	$\{1\}$	$\{0\}$	$\{-3,-2,-1\}$	$\{-3,-2,-1,0,1\}$	...
Candidate for $\mathbf{S}[k]$	2 or 3	-5	2 or 3	0 or 1	-4 or -3	-4	...

We can see that for any recovered sign variation of f_{ℓ_1,ℓ_2,ℓ_3}, there is

1. either one possible value a for $\mathbf{S}[k]$, which leads to a *perfect hint* (diagonal)

$$\mathbf{S}[k] = a;$$

2. or two possible consecutive values $(a, a+1)$ for $\mathbf{S}[k]$, which leads to an *approximate hint* (diagonal)
$$\mathbf{S}[k] + e = a + 0.5 \text{ where } e \sim N_1(0, 1/12).$$

It only remains to apply the tool of Chapter 5 to estimate the new security of the scheme with the knowledge of the secret. We estimate around 1000 approximate hints and 20 perfect ones. By using our tool, the expected bikz security becomes 660, which corresponds to going from 256 bits of security to roughly 175 bits of security. The number of queries is the same as estimated in [Bau+19] : $18,500$.

5.6.3 Hints from structural design

LAC is a Ring-LWE round two candidate of the NIST post-quantum competition [Lu+19]. The secrets are two polynomials $\mathbf{s}_0, \mathbf{s}_1$ (denoted $\mathbf{s}$ and $\mathbf{e}$ in the specifications) whose coefficients follow a distribution $\psi^{n,h}$, the uniform distribution over ternary vectors $\{-1, 0, 1\}^n$ with exactly $h/2$ ones and $h/2$ minus ones. Thus, two structural perfect hints can be derived:
$$\sum_{i=0}^{n-1} \mathbf{s}_0[i] = 0 \text{ and } \sum_{i=0}^{n-1} \mathbf{s}_1[i] = 0.$$

The same structure appears in the submission Round5 (only for $\mathbf{s}_0$) as it also require the number of -1 coefficients to be balanced with number of 1 coefficients of their ternary polynomial. This new knowledge has been included in the security analysis, and the results are stored in Table 5.5. For Round5, we arbitrarily chose for our testing the parameter set R5ND_{1,3,5}KEM_0d. A structure also appears in the NTRU submission [Zha+19], our framework can be applied but there are some subtleties due to the underlying ring and the homogenous property of the underlying BDD problem. We refer to our paper for further details on NTRU [Dac+20].

Remark 28. *Note, however, that integrating such hints removes some q-vectors from the lattice. For LAC, we note that while q-vectors are not in the lattice, a difference of 2 such vectors is still in it, for example, the short vector hint $(q, -q, 0, 0, \ldots, 0) \in \Lambda$. We iteratively integrate $(q, -q, 0, 0, \ldots, 0)$, $(0, q, -q, 0, \ldots, 0)$, $(0, 0, q, -q, \ldots, 0)$, ... until such hints are not worthy anymore, i.e. until such hints do not decrease the cost of the attack anymore.*

Remark 29. *A similar structure is present in the candidate NTRU-Prime in its streamlined and LPR versions [Ber+19]. In the secret vector, the number of ± 1's is fixed to an integer w without knowing the exact number of positive and negative ones. Thus, one can include a modular hint*
$$\sum_{i=0}^{n-1} \mathbf{s}_0[i] = w \mod 2.$$

The loss of security is, however, essentially negligible.

5.7 Perspectives

Perspective 9 (Many other side-channel applications)**.** *The applications showed in this frameworks can exemplify our tool. However, there is much left to do on the application and side-channel side. Among the interesting perspectives, one can cite a power analysis on the NTT computation or the application in a cold-boot scenario.*

Table 5.5: *New security estimates in bikz*

	LAC-128	LAC-192	LAC-256
without hints	509.03	985.64	1104.83
with 2 hints	505.94	982.74	1101.61
	R5ND_{1}KEM_0d	R5ND_{3}KEM_0d	R5ND_{5}KEM_0d
without hints	494.39	658.67	877.71
with 1 hint	492.94	657.23	876.24

Perspective 10 (Adaptation of this tool to the dual attack)**.** *The dual attack is a distinguishing attack that is also part of the security analysis of several schemes. For most NIST schemes it is significantly less efficient than the primal attack, but adapting this work for this attack is still a natural question. In principle, the dual attack could also be made to work within our framework. However, we see two difficulties. First, the cost of the dual attack can not be analyzed solely based on the blocksize of SVP oracle calls. Secondly, the cost of the dual attack has not been confirmed by extensive experiments, unlike the primal one.*

Bibliography

[AAB19] F. Arute, K. Arya, and R. Babbush et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature 574*. Ed. by Joe P. Buhler. 2019, pp. 505–510 *Cited on page 4.*

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. "Efficient Lattice (H)IBE in the Standard Model". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 553–572 *Cited on page 25.*

[Abd+02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. "From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 418–433 *Cited on page 81.*

[ADP18a] Martin R. Albrecht, Amit Deo, and Kenneth G. Paterson. "Cold Boot Attacks on Ring and Module LWE Keys Under the NTT". In: *IACR TCHES* 2018.3 (2018), pp. 173–213 *Cited on page 15.*

[ADP18b] Martin R. Albrecht, Amit Deo, and Kenneth G. Paterson. "Cold Boot Attacks on Ring and Module LWE Keys Under the NTT". In: *IACR TCHES*. Vol. 2018. 3. Aug. 2018, pp. 173–213 *Cited on page 136.*

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108 *Cited on pages 9, 10.*

[Ajt98] Miklós Ajtai. "The Shortest Vector Problem in L2 is NP-hard for Randomized Reductions (Extended Abstract)". In: *30th ACM STOC*. ACM Press, May 1998, pp. 10–19 *Cited on page 9.*

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. "A sieve algorithm for the shortest lattice vector problem". In: *33rd ACM STOC*. ACM Press, July 2001, pp. 601–610 *Cited on page 10.*

[Alb+17] Martin R Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. "Revisiting the expected cost of solving uSVP and applications to LWE". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 297–322 *Cited on pages 10, 128, 142, 144, 145, 154.*

[Alb+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. "Estimate All the LWE, NTRU Schemes!" In: *SCN 18*. Ed. by Dario Catalano and Roberto De Prisco. Vol. 11035. LNCS. Springer, Heidelberg, Sept. 2018, pp. 351–367 *Cited on pages 120, 142, 145.*

[Alk+15] Erdem Alkim, Nina Bindel, Johannes Buchmann, and Özgür Dagdelen. *TESLA: Tightly-Secure Efficient Signatures from Standard Lattices*. Cryptology ePrint Archive, Report 2015/755. 2015 *Cited on page 14.*

[Alk+16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. *NewHope without reconciliation*. Cryptology ePrint Archive, Report 2016/1157. 2016 *Cited on pages 14, 113.*

[Alk+16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum key exchange—a new hope". In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 327–343 *Cited on pages 10, 14, 107, 113, 119, 120, 128, 145.*

[Alk+17] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. "Revisiting TESLA in the Quantum Random Oracle Model". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*. Ed. by Tanja Lange and Tsuyoshi Takagi. Springer, Heidelberg, 2017, pp. 143–162 *Cited on page 14.*

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of learning with errors". In: *JMC* (2015) *Cited on page 128.*

[AR04] Dorit Aharonov and Oded Regev. "Lattice Problems in NP cap coNP". In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 362–371 *Cited on page 8.*

[Aro+93] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. "The Hardness of Approximate Optimia in Lattices, Codes, and Systems of Linear Equations". In: *34th FOCS*. IEEE Computer Society Press, Nov. 1993, pp. 724–733 *Cited on page 9.*

[Baa+19] Hayo Baan, Sauvik Bhattacharya, Scott R. Fluhrer, Óscar García-Morchón, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O. Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. "Round5: Compact and Fast Post-quantum Public-Key Encryption". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Springer, Heidelberg, 2019, pp. 83–102 *Cited on pages 14, 120.*

[Bab85] László Babai. "On Lovász' Lattice Reduction and the Nearest Lattice Point Problem (Shortened Version)". In: *STACS*. 1985 *Cited on page 9.*

[Băe+19] Ciprian Băetu, F Betül Durak, Loıs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. "Misuse Attacks on Post-quantum Cryptosystems". In: *EUROCRYPT*. Springer. 2019 *Cited on page 111.*

[Bai+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". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 3–24 *Cited on pages 27, 28.*

[Bar+16] Gilles Barthe, Sonia Belaıd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 116–129 *Cited on pages 66, 68, 71, 72, 77.*

[Bar+18] Gilles Barthe, Sonia Belaıd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. "Masking the GLP Lattice-Based Signature Scheme at Any Order". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, Apr. 2018, pp. 354–384 *Cited on pages 16, 62–65, 76, 83, 88, 184.*

[Bar+19a] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. *GALACTICS: Gaussian Sampling for Lattice-Based Constant-Time Implementation of Cryptographic Signatures, Revisited*. Cryptology ePrint Archive, Report 2019/511. 2019 *Cited on pages 17, 23, 34, 63–65.*

[Bar+19b] Gilles Barthes, Sonia Belaïd, Thomas Espitau, Mélissa Rossi, and Mehdi Tibouchi. *GALACTICS implementations*. 2019 *Cited on page 39.*

[Bau+19] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. "Assessment of the Key-Reuse Resilience of NewHope". In: *CT-RSA 2019*. Ed. by Mitsuru Matsui. Vol. 11405. LNCS. Springer, Heidelberg, Mar. 2019, pp. 272–292 *Cited on pages 18, 107, 112, 118, 158–160.*

[BB03] David Brumley and Dan Boneh. "Remote Timing Attacks Are Practical". In: *USENIX Security 2003*. USENIX Association, Aug. 2003 *Cited on page 6.*

[BC18] Nicolas Brisebarre and Sylvain Chevillard. "Efficient polynomial L^∞-approximations". In: *18th IEEE Symposium on Computer Arithmetic (ARITH 18)*. IEEE. 2018, pp. 169–176 *Cited on page 35.*

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997), pp. 235–265 *Cited on page 118.*

[Ber+17a] Daniel J. Bernstein, Leon Groot Bruinderink, Tanja Lange, and Lorenz Panny. *HILA5 Pindakaas: On the CCA security of lattice-based encryption with error correction*. Cryptology ePrint Archive, Report 2017/1214. 2017 *Cited on page 111.*

[Ber+17b] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. "NTRU Prime: Reducing Attack Surface at Low Cost". In: *SAC 2017*. Ed. by Carlisle Adams and Jan Camenisch. Vol. 10719. LNCS. Springer, Heidelberg, Aug. 2017, pp. 235–260 *Cited on page 14.*

[Ber+18] Pauline Bert, Pierre-Alain Fouque, Adeline Roux-Langlois, and Mohamed Sabt. "Practical Implementation of Ring-SIS/LWE Based Signature and IBE". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Springer, Heidelberg, 2018, pp. 271–291 *Cited on page 60.*

[Ber+19] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. *PQC Round-2 candidate: NTRU Prime*. Tech. rep. NIST, 2019 *Cited on pages 14, 160.*

[Ber05] Daniel J. Bernstein. *Cache-timing attacks on AES*. Tech. rep. 2005 *Cited on page 6.*

[BG14] Shi Bai and Steven D. Galbraith. "An Improved Compression Technique for Signatures Based on Learning with Errors". In: *CT-RSA 2014*. Ed. by Josh Benaloh. Vol. 8366. LNCS. Springer, Heidelberg, Feb. 2014, pp. 28–47 *Cited on pages 13, 94, 185, 187.*

[Bin+17] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. *qTESLA*. Tech. rep. National Institute of Standards and Technology, 2017 *Cited on page 94.*

[Bin+19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. *qTESLA*. Tech. rep. National Institute of Standards and Technology, 2019 *Cited on pages 14, 15, 25, 50, 63, 94.*

[BMW19] Shi Bai, Shaun Miller, and Weiqiang Wen. *A refined analysis of the cost for solving LWE via uSVP*. Cryptology ePrint Archive, Report 2019/502. 2019 *Cited on page 144.*

[Bo16] Daniel J. Bernstein and VAMPIRE Lab others. *System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives*. https://bench.cr.yp.to/supercop.html. 2016 *Cited on page 51.*

[Bol+19] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. "Order-LWE and the Hardness of Ring-LWE with Entropic Secrets". In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. 2019, pp. 91–120 *Cited on page 148.*

[Boo+18] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. "LWE Without Modular Reduction and Improved Side-Channel Attacks Against BLISS". In: *ASIACRYPT 2018, Part I*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Heidelberg, Dec. 2018, pp. 494–524 *Cited on pages 25, 136.*

[Bos+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. "Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE". In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 1006–1018 *Cited on pages 12, 14, 60, 154.*

[Bos+18a] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. "Assessing the Feasibility of Single Trace Power Analysis of Frodo". In: *SAC*. 2018 *Cited on pages 15, 136, 137, 139, 147, 151, 154–156.*

[Bos+18b] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. *Fly, you fool! Faster Frodo for the ARM Cortex-M4*. Cryptology ePrint Archive, Report 2018/1116. 2018 *Cited on page 102.*

[Bra+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584 *Cited on page 10.*

[Bru+16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. "Flush, Gauss, and Reload - A Cache Attack on the BLISS Lattice-Based Signature Scheme". In: *CHES 2016*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. LNCS. Springer, Heidelberg, Aug. 2016, pp. 323–345 *Cited on pages 15, 25, 47, 63, 136, 148.*

[BV18] David Blackman and Sebastiano Vigna. "Scrambled Linear Pseudorandom Number Generators". In: *CoRR* abs/1805.01407 (2018). arXiv: 1805.01407 *Cited on page 102.*

[Cas+10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. "Bonsai Trees, or How to Delegate a Lattice Basis". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 523–552 *Cited on page 25.*

[CFJ13] Tony Cai, Jianqing Fan, and Tiefeng Jiang. "Distributions of Angles in Random Packing on Spheres". In: *Journal of Machine Learning Research* 14.21 (2013), pp. 1837–1864 *Cited on page 124.*

[CGM19] Yilei Chen, Nicholas Genise, and Pratyay Mukherjee. "Approximate Trapdoors for Lattices and Smaller Hash-and-Sign Signatures". In: *ASIACRYPT 2019*. 2019, pp. 3–32 *Cited on page 60.*

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. "Secure Conversion between Boolean and Arithmetic Masking of Any Order". In: *CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. LNCS. Springer, Heidelberg, Sept. 2014, pp. 188–205 *Cited on pages 62, 69, 70, 179, 182.*

[Cha+99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 398–412 *Cited on pages 6, 64, 65.*

[Che+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography*. National Institute of Standards and Technology (NIST), NISTIR 8105 Draft, U.S. Department of Commerce. Feb. 2016 *Cited on page 4.*

[Che+18] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. "Lizard: Cut off the tail! A practical post-quantum public-key encryption from LWE and LWR". In: *International Conference on Security and Cryptography for Networks*. Springer. 2018, pp. 160–177 *Cited on page 137.*

[Cho17] Arjun Chopra. *GLYPH: A New Instantiation of the GLP Digital Signature Scheme*. Cryptology ePrint Archive, Report 2017/766. 2017 *Cited on pages 80, 89.*

[CMP18] Laurent Castelnovi, Ange Martinelli, and Thomas Prest. "Grafting Trees: A Fault Attack Against the SPHINCS Framework". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Springer, Heidelberg, 2018, pp. 165–184 *Cited on page 15.*

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 1–20 *Cited on pages 10, 128.*

[Cor+14] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. "Higher-Order Side Channel Security and Mask Refreshing". In: *FSE 2013*. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, Heidelberg, Mar. 2014, pp. 410–424 *Cited on page 66.*

[Cor+15] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala. "Conversion from Arithmetic to Boolean Masking with Logarithmic Complexity". In: *FSE 2015*. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, Heidelberg, Mar. 2015, pp. 130–149 *Cited on page 75.*

[Cor14] Jean-Sébastien Coron. "Higher Order Masking of Look-Up Tables". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 441–458 *Cited on pages 69, 72.*

[Cor17] Jean-Sébastien Coron. "High-Order Conversion from Boolean to Arithmetic Masking". In: *CHES 2017*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. LNCS. Springer, Heidelberg, Sept. 2017, pp. 93–114 *Cited on page 16.*

[Cou+18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. "On the Concrete Security of Goldreich's Pseudorandom Generator". In: *ASIACRYPT 2018, Part II*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11273. LNCS. Springer, Heidelberg, Dec. 2018, pp. 96–124 *Cited on page 19.*

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *CHES 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. LNCS. Springer, Heidelberg, Aug. 2003, pp. 13–28 *Cited on page 155.*

[CRW04] George Casella, Christian P. Robert, and Martin T. Wells. "Generalized Accept-Reject sampling schemes". In: *A Festschrift for Herman Rubin*. Ed. by Anirban DasGupta. Vol. Volume 45. Lecture Notes–Monograph Series. Beachwood, Ohio, USA: Institute of Mathematical Statistics, 2004, pp. 342–347 *Cited on page 32.*

[Dac+20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. "LWE with Side Information: Attacks and Concrete Security Estimation". In: *CRYPTO*. 2020 *Cited on pages 18, 135, 141, 146, 152, 158, 160.*

[DAn+19a] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. "Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes". In: *PKC 2019, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. LNCS. Springer, Heidelberg, Apr. 2019, pp. 565–598 *Cited on pages 16, 107, 112, 120, 121, 123, 127, 130, 158.*

[DAn+19b] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. *SABER*. Tech. rep. National Institute of Standards and Technology, 2019 *Cited on pages 14, 119, 120.*

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 423–440 *Cited on page 65.*

[Den03] Alexander W Dent. "A designer's guide to KEMs". In: *IMACC*. Springer. 2003 *Cited on page 110.*

[Dev86] Luc Devroye. *Non-Uniform Random Variate Generation(originally published with)*. New York: Springer-Verlag, 1986 *Cited on page 31.*

[DH76] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* (1976) *Cited on pages 3, 110.*

[Din12] Jintai Ding. *New cryptographic constructions using generalized learning with errors problem*. Cryptology ePrint Archive, Report 2012/387. 2012 *Cited on page 110.*

[DL13] Léo Ducas and Tancrède Lepoint. *BLISS: Bimodal Lattice Signature Schemes*. 2013 *Cited on page 52.*

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. "Efficient Identity-Based Encryption over NTRU Lattices". In: *ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, Heidelberg, Dec. 2014, pp. 22–41 *Cited on pages 25, 60.*

[DN12a] Léo Ducas and Phong Q. Nguyen. "Faster Gaussian Lattice Sampling Using Lazy Floating-Point Arithmetic". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 415–432 *Cited on page 30.*

[DN12b] Léo Ducas and Phong Q. Nguyen. "Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 433–450 *Cited on page 12.*

[DRV20] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. "(One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes". In: *EUROCRYPT*. 2020 *Cited on pages 18, 108, 112, 118, 126–128, 130.*

[Duc+13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. "Lattice Signatures and Bimodal Gaussians". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 40–56 *Cited on pages 13, 14, 23, 24, 32, 40, 48, 50, 63.*

[Duc+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: *IACR TCHES* 2018.1 (2018), pp. 238–268 *Cited on pages 14, 25, 52, 94.*

[DVV18] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. "On the impact of decryption failures on the security of LWE/LWR based schemes." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 1089 *Cited on pages 111, 128, 137, 158, 159.*

[ElG85] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472 *Cited on page 110.*

[Esp+16] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. "Loop-Abort Faults on Lattice-Based Fiat-Shamir and Hash-and-Sign Signatures". In: *SAC 2016*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. LNCS. Springer, Heidelberg, Aug. 2016, pp. 140–158 *Cited on page 63.*

[Esp+17] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. "Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing against strongSwan and Electromagnetic Emanations in Microcontrollers". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, Oct. 2017, pp. 1857–1874 *Cited on pages 23, 25, 63.*

[Est60] Gerald Estrin. "Organization of Computer Systems: The Fixed Plus Variable Structure Computer". In: *Papers Presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference*. IRE-AIEE-ACM '60 (Western). San Francisco, California: ACM, 1960, pp. 33–40 *Cited on page 57.*

[Fac+18] Adrien Facon, Sylvain Guilley, Matthieu Lec'Hvien, Alexander Schaub, and Youssef Souissi. "Detecting cache-timing vulnerabilities in post-quantum cryptography algorithms". In: *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*. IEEE. 2018, pp. 7–12 *Cited on page 60.*

[Flu16] Scott Fluhrer. *Cryptanalysis of ring-LWE based key exchange with key share reuse*. Cryptology ePrint Archive, Report 2016/085. 2016 *Cited on pages 15, 107, 111.*

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Journal of Cryptology* 26.1 (Jan. 2013) *Cited on pages 110, 119.*

[Fou+19] Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. *Key Recovery from Gram-Schmidt Norm Leakage in Hash-and-Sign Signatures over NTRU Lattices*. Cryptology ePrint Archive, Report 2019/1180. 2019 *Cited on page 58.*

[FP85] U. Fincke and M. Pohst. "Improved methods for calculating vectors of short lengthin a lattice, including a complexity analysis". In: *Math. Comp.* (1985), 44(170):463–471 *Cited on page 10.*

[Gar+19] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, Hayo Baan, Markku-Juhani O. Saarinen, Scott Fluhrer, Thijs Laarhoven, and Rachel Player. *Round5*. Tech. rep. NIST, 2019 *Cited on pages 14, 137.*

[GG98] Oded Goldreich and Shafi Goldwasser. "On the Limits of Non-Approximability of Lattice Problems". In: *30th ACM STOC*. ACM Press, May 1998, pp. 1–9 *Cited on page 8.*

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. "Public-Key Cryptosystems from Lattice Reduction Problems". In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Heidelberg, Aug. 1997, pp. 112–131 *Cited on page 12.*

[GJN] Qian Guo, Thomas Johansson, and Alexander Nilsson. *A Generic Attack on Lattice-based Schemes using Decryption Errors with Application to ss-ntru-pke*. IACR ePrint 2019/043 *Cited on page 132.*

[GJN19] Qian Guo, Thomas Johansson, and Alexander Nilsson. *A Generic Attack on Lattice-based Schemes using Decryption Errors*. Cryptology ePrint Archive, Report 2019/043. 2019 *Cited on pages 111, 128, 131.*

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. "Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems". In: *CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. LNCS. Springer, Heidelberg, Sept. 2012, pp. 530–547 *Cited on pages 13–16, 62, 79–81.*

[GM18] Nicholas Genise and Daniele Micciancio. "Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Heidelberg, Apr. 2018, pp. 174–203 *Cited on page 60.*

[GN07] Nicolas Gama and Phong Q Nguyen. "New Chosen-Ciphertext Attacks on NTRU". In: *Public Key Cryptography – PKC 2007*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007 *Cited on page 111.*

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 31–51 *Cited on page 10.*

[Gou01] Louis Goubin. "A Sound Method for Switching between Boolean and Arithmetic Masking". In: *CHES 2001*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. LNCS. Springer, Heidelberg, May 2001, pp. 3–15 *Cited on page 69.*

[GP18] Leon Groot Bruinderink and Peter Pessl. "Differential Fault Attacks on Deterministic Lattice Signatures". In: *IACR TCHES*. Vol. 2018. 3. Aug. 2018, pp. 21–43 *Cited on pages 15, 97, 136.*

[GP99] Louis Goubin and Jacques Patarin. "DES and Differential Power Analysis (The "Duplication" Method)". In: *CHES'99*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. LNCS. Springer, Heidelberg, Aug. 1999, pp. 158–172 *Cited on pages 6, 64.*

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206 *Cited on pages 12, 25.*

[GR19] François Gérard and Mélissa Rossi. *An Efficient and Provable Masked Implementation of qTESLA*. Cryptology ePrint Archive, Report 2019/606. 2019 *Cited on pages 16, 63, 64, 94, 184.*

[Gro96] Lov K Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. New York, NY, USA: ACM, 1996 *Cited on page 6.*

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: (2017) *Cited on page 110.*

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second. SIAM, 2002 *Cited on page 43.*

[HLS18] Andreas Hülsing, Tanja Lange, and Kit Smeets. "Rounded Gaussians - Fast and Secure Constant-Time Sampling for Lattice-Based Crypto". In: *PKC 2018, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 728–757 *Cited on page 30.*

[Hof+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. "NTRUSIGN: Digital Signatures Using the NTRU Lattice". In: *CT-RSA 2003*. Ed. by Marc Joye. Vol. 2612. LNCS. Springer, Heidelberg, Apr. 2003, pp. 122–140 *Cited on page 12.*

[Hof+09] Jeff Hoffstein, Nick Howgrave-Graham, Jill Pipher, and William Whyte. "Practical lattice-based cryptography: NTRUEncrypt and NTRUSign". In: *The LLL Algorithm*. Springer, 2009, pp. 349–390 *Cited on page 137.*

[How+03] Nick Howgrave-Graham, Phong Q Nguyen, David Pointcheval, John Proos, Joseph H Silverman, Ari Singer, and William Whyte. "The Impact of Decryption Failures on the Security of NTRU Encryption". In: (2003) *Cited on page 111.*

[How+20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. "Isochronous Gaussian Sampling: From Inception to Implementation". In: 2020 *Cited on pages 17, 24, 52, 54, 58.*

[How07] Nick Howgrave-Graham. "A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU". In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Aug. 2007, pp. 150–169 *Cited on page 145.*

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NSS: An NTRU Lattice-Based Signature Scheme". In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 211–228 *Cited on page 12.*

[HPS11a] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. "Algorithms for the Shortest and Closest Lattice Vector Problems". In: (2011). Ed. by Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, pp. 159–190 *Cited on page 10.*

[HPS11b] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. "Analyzing Blockwise Lattice Algorithms Using Dynamical Systems". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 447–464 *Cited on page 10.*

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A ring-based public key cryptosystem". In: *Algorithmic Number Theory*. Ed. by Joe P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288 *Cited on page 12.*

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 463–481 *Cited on pages 16, 65, 68, 69, 72.*

[Jia+17] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. *Post-quantum IND-CCA-secure KEM without Additional Hash*. IACR ePrint 2017/1096. 2017 *Cited on page 110.*

[JJ00] Éliane Jaulmes and Antoine Joux. "A Chosen-Ciphertext Attack against NTRU". In: *CRYPTO 2000*. 2000 *Cited on page 111.*

[Kan83] Ravi Kannan. "Improved Algorithms for Integer Programming and Related Lattice Problems". In: *15th ACM STOC*. ACM Press, Apr. 1983, pp. 193–206 *Cited on page 10.*

[Kan87] Ravi Kannan. "Minkowski's convex body theorem and integer programming". In: *Mathematics of operations research*. Vol. 12. 3. INFORMS, 1987, pp. 415–440 *Cited on pages 137, 142.*

[Kar+18] A. Karmakar, S. S. Roy, O. Reparaz, F. Vercauteren, and I. Verbauwhede. "Constant-Time Discrete Gaussian Sampling". In: *IEEE Transactions on Computers* 67.11 (Nov. 2018), pp. 1561–1571 *Cited on page 30.*

[Kar+19] Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. "Pushing the speed limit of constant-time discrete Gaussian sampling. A case study on the Falcon signature scheme". In: *Proceedings of the 56th Annual Design Automation Conference 2019*. 2019, pp. 1–6 *Cited on pages 59, 60.*

[Kel20] Kelenner. *Message on StackExchange*. Accessed on March 2020 *Cited on page 36.*

[Kha+18] Ayesha Khalid, James Howe, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. "Compact, scalable, and efficient discrete Gaussian samplers for lattice-based cryptography". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2018, pp. 1–5 *Cited on page 60.*

[Kho04] Subhash Khot. "Hardness of Approximating the Shortest Vector Problem in Lattices". In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 126–135 *Cited on page 8.*

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 388–397 *Cited on page 6.*

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2014 *Cited on pages 109, 110.*

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 104–113 *Cited on page 6.*

[Len81] A. K. Lenstra. "Lattices and Factorization of Polynomials". In: *SIGSAM Bull.* 15.3 (Aug. 1981), pp. 15–16 *Cited on page 9.*

[Lip+18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. "Meltdown: Reading Kernel Memory from User Space". In: *USENIX Security 2018*. Ed. by William Enck and Adrienne Porter Felt. USENIX Association, Aug. 2018, pp. 973–990 *Cited on page 6.*

[Liu19] Li-Ping Liu. *Linear Transformation of Multivariate Normal Distribution: Marginal, Joint and Posterior*. Accessed on September 2019 *Cited on pages 140, 150.*

[LLL82] Arjen Lenstra, H. Lenstra, and László Lovász. "Factoring Polynomials with Rational Coefficients". In: *Mathematische Annalen* 261 (Dec. 1982) *Cited on pages 8–10.*

[LM06] Vadim Lyubashevsky and Daniele Micciancio. "Generalized Compact Knapsacks Are Collision Resistant". In: *ICALP 2006, Part II*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Vol. 4052. LNCS. Springer, Heidelberg, July 2006, pp. 144–155 *Cited on page 12.*

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. LNCS. Springer, Heidelberg, Feb. 2011, pp. 319–339 *Cited on pages 12, 151.*

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 1–23 *Cited on pages 11, 12, 107, 110, 119, 120.*

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "A Toolkit for Ring-LWE Cryptography". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 35–54 *Cited on page 12.*

[LS12a] Adeline Langlois and Damien Stehlé. *Hardness of decision (R)LWE for any modulus*. Cryptology ePrint Archive, Report 2012/091. 2012 *Cited on page 97.*

[LS12b] Adeline Langlois and Damien Stehlé. *Worst-Case to Average-Case Reductions for Module Lattices*. Cryptology ePrint Archive, Report 2012/090. 2012 *Cited on page 11.*

[Lu+19] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. *PQC Round-2 candidate: LAC*. Tech. rep. NIST, 2019 *Cited on pages 14, 120, 133, 137, 145, 160.*

[LW15] Vadim Lyubashevsky and Daniel Wichs. "Simple Lattice Trapdoor Sampling from a Broad Class of Distributions". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 716–730 *Cited on page 25.*

[Lyu+19] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-DILITHIUM*. Tech. rep. National Institute of Standards and Technology, 2019 *Cited on pages 14, 15, 25, 79.*

[Lyu09] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 598–616 *Cited on pages 13, 40.*

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures without Trapdoors". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 738–755 *Cited on pages 13, 32, 33, 40, 94.*

[Mar13] Jacques Martinet. *Perfect lattices in Euclidean spaces*. Vol. 327. Springer, 2013 *Cited on page 141.*

[Mic20] Daniele Micciancio. *Lattices Algorithms and Applications (course CSE206A)*. Accessed on May 2020 *Cited on page 141.*

[Mig+19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. "Masking Dilithium - Efficient Implementation and Side-Channel Evaluation". In: *ACNS 19*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung. Vol. 11464. LNCS. Springer, Heidelberg, June 2019, pp. 344–362 *Cited on pages 15, 16, 89, 96, 103, 184.*

[MM10] Luca Martino and Joaquín Míguez. "Generalized rejection sampling schemes and applications in signal processing". In: *Signal Processing* 90.11 (2010), pp. 2981–2995 *Cited on page 32.*

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Jan. 2007 *Cited on page 6.*

[MOW17] David McCann, Elisabeth Oswald, and Carolyn Whitnall. "Towards Practical Tools for Side Channel Aware Software Engineering: 'Grey Box' Modelling for Instruction Leakages". In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 199–216 *Cited on pages 154, 155.*

[MP12] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 700–718 *Cited on pages 25, 60.*

[MR07] Daniele Micciancio and Oded Regev. "Worst-case to average-case reductions based on Gaussian measures". In: vol. 37. 1. SIAM, 2007, pp. 267–302 *Cited on pages 11, 149.*

[MRa+99] David M'Raïhi, David Naccache, David Pointcheval, and Serge Vaudenay. "Computational Alternatives to Random Number Generators". In: *SAC 1998*. Ed. by Stafford E. Tavares and Henk Meijer. Vol. 1556. LNCS. Springer, Heidelberg, Aug. 1999, pp. 72–80 *Cited on page 97.*

[MW15] Daniele Micciancio and Michael Walter. "Fast Lattice Point Enumeration with Minimal Overhead". In: *26th SODA*. Ed. by Piotr Indyk. ACM-SIAM, Jan. 2015, pp. 276–294 *Cited on page 10.*

[MW17] Daniele Micciancio and Michael Walter. "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 455–485 *Cited on pages 30, 32, 48, 50, 60, 78.*

[Nae+17] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. *FrodoKEM*. Tech. rep. National Institute of Standards and Technology, 2017 *Cited on page 154.*

[Nae+19] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. *FrodoKEM*. Tech. rep. NIST, 2019 *Cited on pages 14, 60, 107, 120, 154.*

[Ngu19] Phong Nguyen. *Giophanthus and *LWR-based submissions*. 2019 *Cited on page 137.*

[NIS16] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016 *Cited on page 109.*

[NR06] Phong Q. Nguyen and Oded Regev. "Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 271–288 *Cited on page 12.*

[PBY17] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. "To BLISS-B or not to be: Attacking strongSwan's Implementation of Post-Quantum Signatures". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, Oct. 2017, pp. 1843–1855 *Cited on pages 25, 63.*

[PDG14] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. "Enhanced Lattice-Based Signatures on Reconfigurable Hardware". In: *CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. LNCS. Springer, Heidelberg, Sept. 2014, pp. 353–370 *Cited on pages 13, 24, 31, 32, 40, 50, 60.*

[Pea14] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014 *Cited on page 126.*

[Pei10] Chris Peikert. "An Efficient and Parallel Gaussian Sampler for Lattices". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 80–97 *Cited on pages 25, 60.*

[Pei14] Chris Peikert. "Lattice Cryptography for the Internet". In: *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*. Ed. by Michele Mosca. Springer, Heidelberg, Oct. 2014, pp. 197–219 *Cited on pages 107, 110.*

[Pod+17] Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rösler. *Attacking Deterministic Signature Schemes using Fault Attacks*. Cryptology ePrint Archive, Report 2017/1014. 2017 *Cited on page 97.*

[Pöp+19] Thomas Pöppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. *NewHope*. Tech. rep. NIST, 2019 *Cited on pages 14, 107, 113, 114, 120, 148.*

[Por18] Thomas Pornin. *Why Constant-Time Crypto?* Tech. rep. 2018 *Cited on page 6.*

[Por19] Thomas Pornin. *New Efficient, Constant-Time Implementations of Falcon*. Cryptology ePrint Archive, Report 2019/893. 2019 *Cited on page 59.*

[Por20] Thomas Pornin. *BearSSL*. Accessed on March 2020 *Cited on page 27.*

[PR06] Chris Peikert and Alon Rosen. "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Heidelberg, Mar. 2006, pp. 145–166 *Cited on page 12.*

[Pre+19] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. National Institute of Standards and Technology, 2019 *Cited on pages 12, 14, 25, 53.*

[Pre17] Thomas Prest. "Sharper Bounds in Lattice-Based Cryptography Using the Rényi Divergence". In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Heidelberg, Dec. 2017, pp. 347–374 *Cited on pages 27–29, 31, 34.*

[QCD19] Yue Qin, Chi Cheng, and Jintai Ding. *A Complete and Optimized Key Mismatch Attack on NIST Candidate NewHope*. Cryptology ePrint Archive, Report 2019/435. 2019 *Cited on page 118.*

[Rav+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. *Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate*. Cryptology ePrint Archive, Report 2018/821. 2018 *Cited on page 136.*

[Rav+19] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. "Exploiting Determinism in Lattice-based Signatures: Practical Fault Attacks on pqm4 Implementations of NIST Candidates". In: *ASIACCS 19*. Ed. by Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang. ACM Press, July 2019, pp. 427–440 *Cited on pages 15, 136.*

[Reg03] Oded Regev. "New lattice based cryptographic constructions". In: *35th ACM STOC*. ACM Press, June 2003, pp. 407–416 *Cited on page 12.*

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93 *Cited on page 10.*

[Reg06] Oded Regev. "Lattice-Based Cryptography (Invited Talk)". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 131–141 *Cited on page 12.*

[Ros+17] Melissa Rossi, Mike Hamburg, Michael Hutter, and Mark E. Marson. "A Side-Channel Assisted Cryptanalytic Attack Against QcBits". In: *CHES 2017*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. LNCS. Springer, Heidelberg, Sept. 2017, pp. 3–23 *Cited on page 19.*

[RP10] Matthieu Rivain and Emmanuel Prouff. "Provably Secure Higher-Order Masking of AES". In: *CHES 2010*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. LNCS. Springer, Heidelberg, Aug. 2010, pp. 413–427 *Cited on pages 66, 68, 72.*

[RS06] Phillip Rogaway and Thomas Shrimpton. "A Provable-Security Treatment of the Key-Wrap Problem". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 373–390 *Cited on page 6.*

[Sch+19] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. *CRYSTALS-KYBER*. Tech. rep. NIST, 2019 *Cited on pages 14, 107, 111, 119, 120, 148.*

[SE94] Claus-Peter Schnorr and Martin Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". In: *Mathematical programming* (1994) *Cited on pages 10, 128.*

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134 *Cited on pages 4, 5.*

[SMC10] S. Chevillard, M. Joldeş, and C. Lauter. "Sollya: An Environment for the Development of Numerical Codes". In: *Mathematical Software - ICMS 2010*. Ed. by K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama. Vol. 6327. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, Sept. 2010, pp. 28–31 *Cited on page 34.*

[Sob63] S.L Sobolev. "On a theorem of functional analysis". In: *Transl. Amer. Math. Soc.* 34 (1963), pp. 39–68 *Cited on page 35.*

[Ste+09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. "Efficient Public Key Encryption Based on Ideal Lattices". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 617–635 *Cited on pages 11, 12.*

[SXY17] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. *Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model*. IACR ePrint 2017/1005. 2017 *Cited on page 110.*

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. "Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms". In: *TCC 2016*. Springer, 2016 *Cited on page 110.*

[TW19] Mehdi Tibouchi and Alexandre Wallet. *One Bit is All It Takes: A Devastating Timing Attack on BLISS's Non-Constant Time Sign Flips*. Cryptology ePrint Archive, Report 2019/898. 2019 *Cited on page 25.*

[Wal19] Michael Walter. "Sampling the Integers with Low Relative Error". In: *AFRICACRYPT 19*. Ed. by Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi. Vol. 11627. LNCS. Springer, Heidelberg, July 2019, pp. 157–180 *Cited on page 31.*

[Wun19] Thomas Wunderer. "A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack". In: *J. Mathematical Cryptology* 13.1 (2019), pp. 1–26 *Cited on page 52.*

[Zha+17] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. *NTRUEncrypt*. Tech. rep. National Institute of Standards and Technology, 2017 *Cited on pages 131, 132.*

[Zha+19] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M. Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, and Oussama Danba. *PQC Round-2 candidate: NTRU*. Tech. rep. NIST, 2019 *Cited on pages 14, 111, 160.*

[ZSS18] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. *FACCT: FAst, Compact, and Constant-Time Discrete Gaussian Sampler over Integers*. Cryptology ePrint Archive, Report 2018/1234. 2018 *Cited on pages 34, 50.*

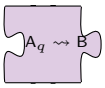
[ZSS19] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. *Compact and Scalable Arbitrary-centered Discrete Gaussian Sampling over Integers*. Cryptology ePrint Archive, Report 2019/1011. 2019 *Cited on pages 59, 60.*

Appendix

A Appendix: Masked Gadgets

▶ Adapting the mask conversions

Using sec_{+_q} instead of sec_+ in the algorithms of [CGV14, Section 4], we also immediately obtain an algorithm denoted $\mathsf{A}_q{\rightsquigarrow}\mathsf{B}$ for converting a mod-q arithmetic masking $x = \sum_{i=0}^{d} x_i \bmod q$ of a value $x \in [0, q)$ into a Boolean masking $z = \bigoplus_{i=0}^{d} z_i$ of the same value. The naive way of doing so (see Gadget 13), which is the counterpart of [CGV14, §4.1], is to simply construct a Boolean masking of each of the shares x_i, and to apply sec_{+_q} to those masked values iteratively. This is simple and secure, but as noted by Coron et al., this approach has cubic complexity in the masking order d (because sec_+ and hence sec_{+_q} are quadratic). A more advanced, recursive approach allows to obtain quadratic complexity for the whole conversion: this is described in [CGV14, Section 4.2], and directly applies to the lattice-based signatures setting. It was not applied in our paper but it is presented here as a contribution of this thesis in Gadget 14. When a masked value composed of an odd number of shares d is presented to the algorithm, it first splits them in two uneven parts of size $\lfloor d/2 \rfloor + 1$ and $\lfloor d/2 \rfloor$ before proceeding to the recursive call. The subroutine Expand takes as input an arbitrary number of shares d' and the bit size of the masks. It expands them in $2d'$ shares (with a 0 padding) and then refreshes them using $\mathsf{Ref}_{\oplus}(., w)$. Applying Expand to both parts, we end up with a part p_1 of size $d+1$ and a part p_2 of size $d-1$. We merge the two last shares of p_1 and append a zero to p_2 to get two size d masking that are finally added together to yield the final boolean masking.



Mask conversion: This gadget denoted $\mathsf{A}_q{\rightsquigarrow}\mathsf{B}$ converts an input $(x_i)_{0\le i\le d}$ in mod q arithmetic masked form to $(z_i)_{0\le i\le d}$ in Boolean masked form such that $(\bigoplus_i z_i) = (\sum_i x_i \mod q)$. It is described in Gadget 13, and a more efficient version is in Gadget 14.

Gadget 13 — Conversion arithmetic to Boolean masking; simple cubic version

Data: Arithmetic masking $(x_i)_{0\le i\le d}$ modulo q of an integer x; the bit size w of the returned masks (with $2^w > 2q$)
Result: A Boolean masking $(z_i)_{0\le i\le d}$ of x

$(z_i)_{0\le i\le d} \coloneqq (0, \ldots, 0)$
for $j = 0$ *to* d **do**
  $(b_i)_{0\le i\le d} \coloneqq (x_j, 0, \ldots, 0)$
  $(b_i)_{0\le i\le d} \leftarrow \mathsf{Ref}_{\oplus}\big((b_i)_{0\le i\le d}, w\big)$
  $(z_i)_{0\le i\le d} \leftarrow \mathsf{sec}_{+_q}\big((z_i)_{0\le i\le d}, (b_i)_{0\le i\le d}, w\big)$
end
return $(z_i)_{0\le i\le d}$

Gadget 14 — Conversion arithmetic to Boolean masking ($\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$)

Data: Arithmetic masking $(x_i)_{0\le i\le d}$ modulo q of an integer x; the bit size w of the returned masks (with $2^w > 2q$)
Result: A Boolean masking $(z_i)_{0\le i\le d}$ of x

1 **if** $d = 0$ **then**
2 $\quad z_0 \coloneqq x_0$
3 $\quad$ **return** z_0
4 **end**
5 $\mathtt{HALF} \coloneqq \lfloor d/2 \rfloor$
6 $(a_i)_{0\le i\le \mathtt{HALF}} \leftarrow \mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}((x_i)_{0\le i\le \mathtt{HALF}})$
7 $(a'_i)_{0\le i\le 2*\mathtt{HALF}} \coloneqq \mathsf{Expand}((a_i)_{0\le i\le \mathtt{HALF}}, w)$
8 $(y_i)_{0\le i\le \lfloor (d-1)/2\rfloor} \leftarrow \mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}((x_i)_{\mathtt{HALF}+1\le i\le d})$
9 $(y'_i)_{0\le i\le 2*\lfloor (d-1)/2\rfloor} \coloneqq \mathsf{Expand}((y_i)_{0\le i\le \lfloor (d+1)/2\rfloor}, w)$
10 **if** *d is even* **then**
11 $\quad y'_{2*\lfloor (d-1)/2\rfloor} \coloneqq 0$
12 $\quad x'_{2*\mathtt{HALF}-1} \coloneqq x'_{2*\mathtt{HALF}-1} \oplus x'_{2*\mathtt{HALF}}$
13 **end**
14 $(z_i)_{0\le i\le d} \leftarrow \mathsf{sec}_{+_q}((x'_i)_{0\le i\le d}, (y'_i)_{0\le i\le d}, w)$

Lemma 23. *Gadget* $\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$ *is* d*-SNI secure.*

Proof: Let us first prove the d-SNI security of Gadget 13. A graphical representation is in Fig. A.1.

Let O be a set of observations performed by the attacker on the final returned value, let I_{A_j} be the set of internal observations made in step j in the gadget sec_{+_q} (line 5), and I_{R_j} be the set of internal observations made in the step j in the initialization of b (line 3) or in the $\mathsf{Ref}_{\oplus}$ (line 4). Assuming that $|O| + \sum(|I_{A_j}| + |I_{R_j}|) \le d$, the gadget is d-SNI secure, if we can build a simulator allowing to simulate all the internal and output observations made by the attacker using a set S of shares of x such that $|S| \le \sum(|I_{A_j}| + |I_{R_j}|)$.

At the last iteration (see Fig. A.2), the set of observations $O \cup I_{A_d}$ can be simulated using a set $S_{z_{d-1}}$ of shares of z and $S_{b_{d-1}}$ of shares of b with $|S_{z_{d-1}}| \le |O| + |I_{A_d}|$ and $|S_{b_{d-1}}| \le |O| + |I_{A_d}|$ (because the gadget sec_{+_q} is d-NI secure). Since the $\mathsf{Ref}_{\oplus}$ is d-SNI secure, the sets $S_{b_{d-1}}$ and I_{R_d} can be simulated using a set $S_{b'_{d-1}}$ of input shares with $|S_{b'_{d-1}}| \le |I_{R_d}|$. If I_{R_d} is not empty, then $S_{b'_{d-1}}$ may contain x_d, so we add x_d to S. For each iteration of the loop, this process can be repeated. At the very first iteration, several shares of z may be necessary to simulate the set of observations. However, there are all initialized to 0, nothing is added to S.

At the end we can conclude that the full algorithm can be simulated using the set S of input shares. Furthermore we have $|S| \le \sum |I_{R_j}|$ (since a_j is added in S only if I_{R_j} is not empty), so we can conclude that $|S| \le \sum |I_{A_j}| + |I_{R_j}|$ which concludes the proof. The security of $\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$ (Gadget 14) is the same. The only difference is that we generalize it for d being arbitrary (i.e. non power of two). This still keeps the d-SNI property. □

▶ The reverse mask conversion

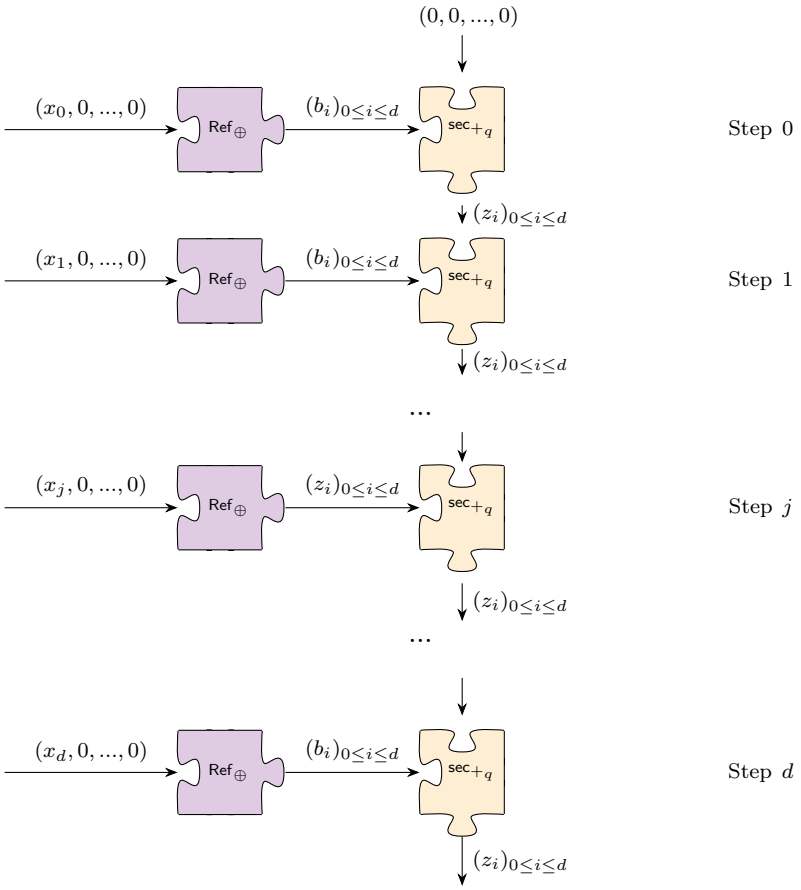


Figure A.1: *Graphical Representation of* $\mathsf{A}_{\hat{q}} \rightsquigarrow \mathsf{B}$

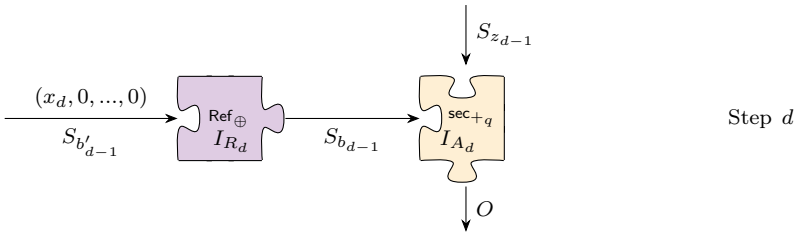


Figure A.2: *Last step of* $\mathsf{A}_{\hat{q}} \rightsquigarrow \mathsf{B}$ *with probes.*

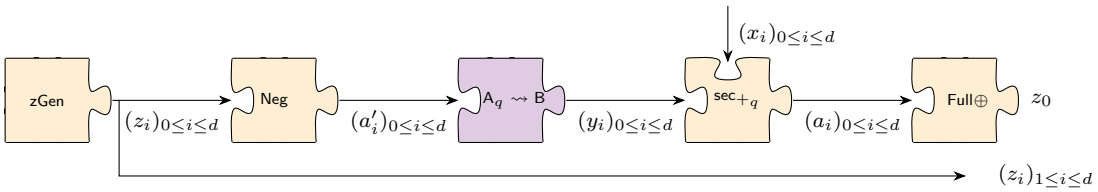
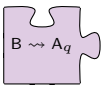


Figure A.3: *Graphical Representation of* $\mathsf{B}\rightsquigarrow\mathsf{A}_q$



Reverse Mask conversion: This gadget denoted $\mathsf{B}\rightsquigarrow\mathsf{A}_q$ converts an input $(x_i)_{0\le i\le d}$ in a Boolean masked form to $(z_i)_{0\le i\le d}$ in mod q arithmetic masked form such that $(\sum_i z_i \mod q) = (\bigoplus_i x_i)$. It is described in Gadget 15.

With both algorithms sec_{+_q} and $\mathsf{A}_q\rightsquigarrow\mathsf{B}$ in hand, we can easily build the $\mathsf{B}\rightsquigarrow\mathsf{A}_q$ algorithm by mimicking [CGV14, Algorithm 6]. To convert the Boolean masking $(x_i)_{0\le i\le d}$ of x to a mod-q arithmetic masking, we first generate random integer shares $z_i \in [0, q)$, $1 \le i \le d$, uniformly at random. We then define $a'_i = -z_i \mod q = q - z_i$ for $1 \le i \le d$ and $a'_0 = 0$. The tuple $(a'_i)_{0\le i\le d}$ is thus a mod-q arithmetic masking of the sum $a' = -\sum_{1\le i\le d} z_i \mod q$. Using $\mathsf{A}_q\rightsquigarrow\mathsf{B}$, we convert this arithmetic masking to a Boolean masking $(y_i)_{0\le i\le d}$, so that $\bigoplus_{i=0}^{d} y_i = a'$. Now, let $(a_i)_{0\le i\le d} = \mathsf{sec}_{+_q}\big((x_i)_{0\le i\le d}, (y_i)_{0\le i\le d}\big)$; this is a Boolean masking of:

$$a = (x + a') \bmod q = \left(x - \sum_{i=1}^{d} z_i\right) \bmod q.$$

We then securely unmask this value using the Full⊕ procedure, and set $z_0 = a \mod q$. Then, we have:

$$\sum_{i=0}^{d} z_i \bmod q = a + \sum_{i=1}^{d} z_i \bmod q = x - \sum_{i=1}^{d} z_i + \sum_{i=1}^{d} z_i \bmod q = x \bmod q.$$

Gadget 15 — Conversion Boolean to arithmetic masking ($\mathsf{B}\rightsquigarrow\mathsf{A}_q$)

Data: Boolean masking $(x_i)_{0\le i\le d}$ of an integer x; the modulus q
Result: A arithmetic masking $(z_i)_{0\le i\le d}$ of x

1 generate uniform integers $(z_i)_{1\le i\le d}$ in $[0, q)$
2 $a'_0 := 0$
3 $a'_i := -z_i \mod q$ for $i = 1, \ldots, d$
4 $(y_i)_{0\le i\le d} \leftarrow \mathsf{A}_q\rightsquigarrow\mathsf{B}\big((a'_i)_{0\le i\le d}\big)$
5 $(a_i)_{0\le i\le d} \leftarrow \mathsf{sec}_{+_q}\big((x_i)_{0\le i\le d}, (y_i)_{0\le i\le d}\big)$
6 $z_0 := \mathsf{Full}\oplus\big((a_i)_{0\le i\le d}\big)$
7 **return** $(z_i)_{0\le i\le d}$

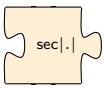
Lemma 24. *Gadget* $\mathsf{B}\rightsquigarrow\mathsf{A}_q$ *is d-SNI secure.*

Proof: Full⊕ is d-NI secure and z_0 is not revealed after its execution. All the $\delta_0 \le d$ observations made by the attacker of this last instance of Full⊕ can be perfectly

simulated with z_0 (for the observations performed after the unobserved linear refreshing) and at most $\delta_0 - 1$ shares of a (for the observations made before the unobserved linear refreshing).

□

▶ AbsVal: Compute an absolute value



Absolute value: This gadget denoted sec|.| takes an input $(x_i)_{0\leq i\leq d}$ in a Boolean masked and returns an output $(z_i)_{0\leq i\leq d}$ in Boolean masked form such that $(\bigoplus_i z_i) = |\bigoplus_i x_i|$. It is described in Gadget 16.

For some masked comparisons, two comparisons on each signed coefficients are necessary. And, sometimes, it is actually less intensive to explicitly compute the absolute value and do only one comparison. The gadget takes as input any integer x masked in Boolean form on w bits and outputs its absolute value (i.e. x if $x < 2^{w-1}$ and $x - 2^w$ if $x \geq 2^{w-1}$). Since computers are performing two's complement arithmetic, the absolute value of x can be computed as follows:

1. $m \coloneqq x \gg RADIX - 1$
2. $|x| \coloneqq (x + m) \oplus m$

We recall that we consider signed integers. So, one can note that the $\gg$ in the first step is an arithmetic shift and actually writes the sign bit in the whole register. If x is negative then $m = -1$ (all ones in the register) and if x is positive then $m = 0$. The gadget sec|.| is using the same technique to compute $|x|$. The small difference is that the sign bit is in position w instead of position RADIX. This is why step 1 is moving the sign bit (modulo 2^w) in first position before extending it to the whole register to compute the mask.

Gadget 16 — Absolute Value (sec|.|)

Data: A boolean masking $(x_i)_{0\leq i\leq d}$ of some integer $x \in \mathbb{Z}$ and an integer w
Result: A boolean masking $(z_i)_{0\leq i\leq d}$ corresponding to the absolute value of x

1 $(mask_i)_{0\leq i\leq d} \coloneqq (x_i)_{0\leq i\leq d} \gg (w-1)$
2 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{Ref}_{\oplus}((x_i)_{0\leq i\leq d})$
3 $(x_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_{+}((x'_i)_{0\leq i\leq d}, (mask_i)_{0\leq i\leq d})$
4 $(z_i)_{0\leq i\leq d} \coloneqq ((x_i)_{0\leq i\leq d} \oplus (mask_i)_{0\leq i\leq d}) \wedge (2^w - 1)$
5 **return** $(z_i)_{0\leq i\leq d}$

For computing the absolute value of an element x modulo q. It consists in calling sec|.|$(x, \lceil \log_2(q) \rceil)$.

Lemma 25. *The gadget* sec|.| *in Gadget 16 is d-NI secure.*

Proof: A graphical representation of sec|.| is in Fig. A.4.

We consider that the attacker made $\delta \leq d$ observations. In the following, we prove that all these δ observations can be perfectly simulated with at most δ shares of $(x_i)_{0\leq i\leq d}$.
In the following, we consider the following distribution of the attacker's δ observations: δ_1 observed during the computations of the shift that produces shares of $(mask_i)_{0\leq i\leq d}$, δ_2 observed during the computations of the $\mathsf{Ref}_{\oplus}$ that produces

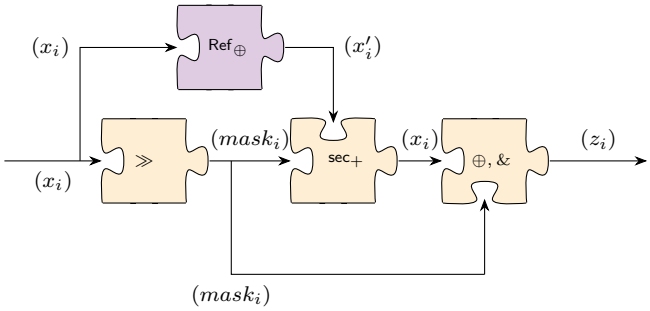


Figure A.4: sec|.| *structure*

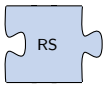
$(x'_i)_{0\leq i\leq d}$, δ_3 observed during the sec$_+$, and δ_4 observed during the final $\oplus$ and $\wedge$ step. Finally, we have $\sum_{i=1}^{4}\delta_i \leq \delta$.

We build the proof classically from right to left. By linearity for Boolean masking, the final $\oplus$ and $\wedge$ step is d-NI. It is also an affine gadget. In other words, each observation can be simulated with either one share of x or one share of $mask$. Thus, all the observations from its call can be simulated with at most δ_4 shares among all the shares of x and $mask$. Then it can be simulated with at most x_1 shares of x and x_2 shares of $mask$ with $x_1 + x_2 = \delta_4$.The gadget sec$_+$ is d-NI then all the observations from its call can be simulated with at most $x_1 + \delta_3$ shares of $mask$ and x'. Identically, the shift is d-NI (by linearity), so the observations from its call can be simulated with at most $\delta_1 + (x_1 + \delta_3) + x_2 = \delta_1 + \delta_3 + \delta_4$ shares of x. Finally, all the observations during the computations of sec|.| can be simulated with at most $\delta_1 + \delta_2 + \delta_3 + \delta_4 \leq \delta$ shares of x.

□

▶ Uniform Rejection Sampling

In many lattice-based Fiat–Shamir with aborts signature schemes, there is a rejection sampling step. Let us consider a uniform one. This check consists in knowing if an arithmetically masked element in $x \in \mathbb{Z}/q\mathbb{Z}$ is in the interval $[-\texttt{param}, \texttt{param}]$. Again, carrying out this check using mod-q arithmetic masking seems difficult, so we again resort to Boolean masking using $\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$.



Uniform Rejection Sampling: This gadget denoted RS-Coeff in its coefficient version and RS in its polynomial version takes an arithmetic mod q input $(x_i)_{0\leq i\leq d}$ and an unmasked value param and outputs 1 iff $|\sum_i x_i \mod q| \leq \texttt{param}$, 0 otherwise. It is presented in Gadget 17 and Gadget 18.

In [Bar+18], we designed a gadget verifying that the centered representative of a masked integer is greater than $-\texttt{param}$ and applied it to both x and $-x$. In [Mig+19], a less computationally intensive approach was taken: their rejection sampling gadget takes as input an arithmetic masking of a coefficient $x \in \mathbb{Z}/q\mathbb{Z}$ identified by its canonical representative and checks directly that either $x - \texttt{param}$ is negative or $x - q + \texttt{param}$ is positive. This can be easily done using precomputed constants $(-\texttt{param} - 1, 0, \ldots, 0)$ and $(-q + \texttt{param}, 0, \ldots, 0)$. The last approach we introduced (in [GR19]) is similar but we

first compute the absolute value of x and perform the masked test $|x| \leq \texttt{param}$. This saves the need for a masked operation to aggregate both tests.

Gadget 17 — Rejection sampling for integers (RS-Coeff)

Data: The shared element $(x_i)_{0\leq i\leq d}$ to check in mod-q arithmetic masked representation; $\texttt{param}$
Result: The bit rs equal to 1 iff $|x| \leq \texttt{param}$, and 0 otherwise.

$(\mathrm{SUP}_i)_{0\leq i\leq d} \coloneqq (-\texttt{param}, 0, ..., 0)$
$(a_i)_{0\leq i\leq d} \leftarrow \mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}((x_i)_{0\leq i\leq d})$
$(a'_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}|.|((a_i)_{0\leq i\leq d}, \lceil \log_2(q) \rceil)$
$(a'_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_{+_q}((a'_i)_{0\leq i\leq d}, (\mathrm{SUP}_i)_{0\leq i\leq d})$
$(b_i)_{0\leq i\leq d} \coloneqq ((a'_i)_{0\leq i\leq d} \gg \mathrm{RADIX} - 1)$
return $rs := \mathsf{Full}\oplus((b_i)_{0\leq i\leq d})$

Lemma 26. *Gadget* RS-*Coeff is* d-NI *secure.*

Proof: The rejection sampling is a succession of gadgets without cycle. Thus, for proving its d-$\texttt{NI}$ security, it remains to prove the d-$\texttt{NIo}$ or d-$\texttt{NI}$ security of each of its gadgets: $\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$, $\mathsf{sec}|.|$, sec_{+_q}, $\gg$ and $\mathsf{Full}\oplus$. As seen before, $\mathsf{A}_q\!\rightsquigarrow\!\mathsf{B}$ (Lemma 23), $\mathsf{sec}|.|$ (Lemma 25), $\mathsf{Full}\oplus$ (Lemma 11) and sec_+ (Lemma 14) are d-$\texttt{NI}$. The $\gg$ is linear for Boolean masking, so it is d-$\texttt{NI}$. Thus, rejection sampling is d-$\texttt{NI}$.

□

The Rejection sampling gadget can be generalized for polynomials in R_q (defined in Eq. (1.3)) by applying RS to each of the coefficient. The values $rs^{(j)}$ correspond to the conditions of rejection, and more precisely, the positions of the coefficients of the polynomial that do not pass the rejections. Such a knowledge does not impact the security of the scheme because the rejection probability does not depend on the position of the coefficients and their number.

Gadget 18 — Rejection sampling (RS)

Data: A shared polynomial $(\mathbf{x}_i)_{0\leq i\leq d}$ to check in mod-q arithmetic masked representation, $\texttt{param}$
Result: The bit rs equal to 1 if all the coefficients, denoted $x^{(i)}$ of $\mathbf{x}$ satisfy $|x^{(i)}| \leq \texttt{param}$, and 0 otherwise.

for $j = 0$ *to* $n-1$ **do**
  $rs^{(j)} \coloneqq \mathsf{RS\text{-}Coeff}(x^{(j)})$
end
return $rs := rs^{(0)} \wedge rs^{(1)} \cdots \wedge rs^{(n-1)}$

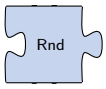
Then, from Remark 5 and Lemma 26, we can make the following corollary.

Corollary 3. *Gadget* RS *is* d-$\texttt{NIo}$ *secure with public outputs* $rs^{(j)}$.

▶ Masked rounding

In certain schemes, a compression technique [BG14] is introduced to reduce the size of the signature. It implies rounding coefficients of a polynomial and keeping the k most significant bits. In many cases, revealing the polynomial before rounding would allow an

adversary to get extra information on secret values and thus, this operation has to be done on the masked polynomial.



Rounding: This gadget denoted Rnd-Coeff in its coefficient version and Rnd in its polynomial version in R_q takes an arithmetic mod q input $(x_i)_{0\leq i\leq d}$ and an unmasked value `Tail` and outputs $z = \lfloor \sum_i x_i \bmod q \rceil_{\texttt{Mst}}$ where $\lfloor\cdot\rceil_{\texttt{Mst}}$ means dropping `Tail` least significant bits. It is presented in Gadget 19 and Gadget 20.

The first step is to compute the centered representative of x, i.e. subtract q from x if $x \geq q/2$. The second step is the computation of $(x - \lfloor x \rceil_{\texttt{Lst}})/2^{\texttt{Tail}}$ (where $\lfloor\cdot\rceil_{\texttt{Lst}}$ defines keeping the `Tail` least significant bits). We used a small trick here. Subtracting the centered representative modulo $2^{\texttt{Tail}}$ is actually equivalent to the application of a rounding to the closest multiple of $2^{\texttt{Tail}}$ with ties rounded down. Hence we first computed $x + 2^{\texttt{Tail}-1} - 1$ and dropped the `Tail` least significant bits. This is analogous to computing $\lfloor x \rceil = \lfloor x + 0.499\ldots \rfloor$ to find the closest integer to a real value.

Gadget 19 — Masked rounding for integers - (Rnd-Coeff)

Data: An arithmetic masking $(x_i)_{0\leq i\leq d}$ of some integer x, `Tail` the number of most significant bits that are kept
Result: An integer z corresponding to the modular rounding of x, i.e. $z = \lfloor x \rceil_{\texttt{Mst}}$ where $\lfloor\cdot\rceil_{\texttt{Mst}}$ denotes dropping the `Tail` least significant bits

1 $(\text{MINUS\_Q\_HALF}_i)_{0\leq i\leq d} := (-q/2 - 1, 0, ..., 0)$
2 $(\text{CONST}_i)_{0\leq i\leq d} := (2^{\texttt{Tail}-1} - 1, 0, ..., 0)$
3 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{A}_q{\rightsquigarrow}\mathsf{B}(x_i)_{0\leq i\leq d}$
4 $(b_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+((x'_i)_{0\leq i\leq d}, (\text{MINUS\_Q\_HALF}_i)_{0\leq i\leq d})$
5 $b_0 = \neg b_0$
6 $(b_i)_{0\leq i\leq d} := ((b_i)_{0\leq i\leq d} \gg \text{RADIX} - 1) \ll \lceil \log_2(q) \rceil$
7 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{Refresh}((x'_i)_{0\leq i\leq d})$
8 $(x'_i)_{0\leq i\leq d} := (x'_i)_{0\leq i\leq d} \oplus (b_i)_{0\leq i\leq d}$
9 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+((x'_i)_{0\leq i\leq d}, (\text{CONST}_i)_{0\leq i\leq d})$
10 $(x'_i)_{0\leq i\leq d} := (x'_i)_{0\leq i\leq d} \gg \texttt{Tail}$
11 **return** $z := \mathsf{Full}{\oplus}((x'_i)_{0\leq i\leq d})$

Gadget 20 — Masked rounding (Rnd)

Data: A shared polynomial $(\mathbf{x}_i)_{0\leq i\leq d}$, in mod-$q$ arithmetic masked representation, `Tail` the number of least significant bits to drop
Result: The polynomial $\mathbf{z}$ corresponding to the modular rounding of $\mathbf{x}$

1 **for** $j = 0$ *to* $n - 1$ **do**
2 $\quad z^{(j)} \leftarrow \mathsf{Rnd\text{-}Coeff}((x_i^{(j)})_{0\leq i\leq d}, \texttt{Tail})$
3 **end**
4 **return** $\mathbf{z} := \sum_i z^{(i)} \cdot X^i$

Lemma 27. *The gadget* Rnd-Coeff *in Gadget 19 is d-NI.*

Proof: A graphical representation of Gadget 19 is in Fig. A.5. Let $\delta \leq d$ be the number of observations made by the attacker. Our goal is to prove that all these δ observations can be perfectly simulated with at most δ shares of $(x_i)_{0\leq i\leq d}$.

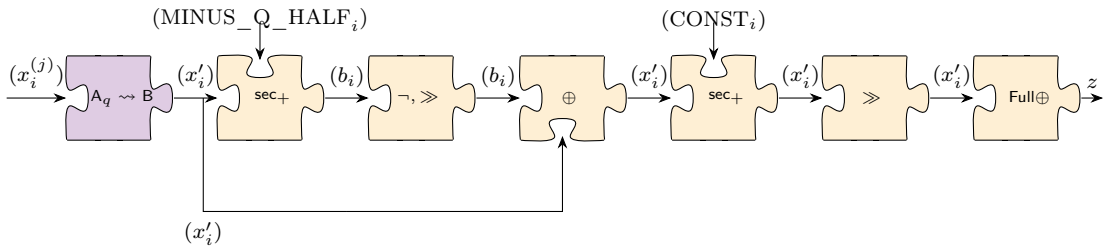


Figure A.5: *Masked Rounding structure* MaskedRound

In the following, a distribution of the attacker's δ observations is considered as follows: δ_1 observed during $\mathsf{A}_q\rightsquigarrow\mathsf{B}$ calculations producing shares of $(x'_i)_{0\leq i\leq d}$, δ_2 observed during sec_+ calculations producing $(b_i)_{0\leq i\leq d}$, δ_3 observed during the switch and shift stages, δ_4 observed during the $\oplus$ operation, δ_5 observed during sec_+ calculations producing $(x'_i)_{0\leq i\leq d}$, δ_6 observed during the final shift stage, and δ_7 observed during the final $\mathsf{Full}\oplus$ stage. Finally, we have $\sum_{i=1}^{7}\delta_i \leq \delta$.

We build the proof from right to left. The algorithm $\mathsf{Full}\oplus$ is d-NI. As a consequence, all the observations from its call can be perfectly simulated with at most $\delta_7 \leq \delta$ shares of x'. The shift algorithm is a linear operation and thus it is d-NI secure. Thus, all observations from its call can be perfectly simulated with at most $\delta_6 + \delta_7 \leq \delta$ shares of x'. The algorithm sec_+ is d-NI secure and similarly, all the observations from its call can be simulated with at most $\delta_5 + \delta_6 + \delta_7 \leq \delta$ shares of x' and $CONST$ (but the latter is a public constant). The $\oplus$ operation is also linear, so it is d-NI. Then, all the observations made from its call can be simulated with at most $\delta_4 + \delta_5 + \delta_6 + \delta_7 \leq \delta$ shares of x' and b, and with the knowledge of u. Actually, we remark that $\oplus$ is also a affine gadget. Thus, all the observations can be exactly simulated with at most v_1 shares of x' and v_2 shares of b such that $v_1 + v_2 = \delta_4 + \delta_5 + \delta_6 + \delta_7$. Let us consider now the switch and shift operations. They are linear so d-NI secure and thus all observations made from its call can be simulated with at most $\delta_3 + v_2 \leq \delta$ observations on b. Considering the first instance of sec_+, its $d-$NI security implies that all the observations from its call can be simulated with at most $\delta_2 + \delta_3 + v_2 \leq \delta$ shares of x' and MINUS_Q_HALF (but the latter is a public constant). Finally, we consider the algorithm $\mathsf{A}_q\rightsquigarrow\mathsf{B}$ which is d-NI secure. There are at most $v_1 + (\delta_2 + \delta_3 + v_2) = \sum_{i=2}^{7}\delta_i$ observations made on the outputs and δ_1 made locally. Thus, all the observations during Gadget 20 can be simulated with at most $\sum_{i=1}^{8}\delta_i \leq \delta \leq d$ shares of the input x.

□

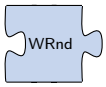
From Remark 5 and Lemma 27, we can make the following corollary.

Corollary 4. *Gadget* Rnd *is d-NI secure, thus d-NIo secure with public outputs $z^{(j)}$.*

▶ Masked well-rounded

In Bai–Galbraith signatures [BG14], the signature scheme can fail the verification and may have to be restarted even if the rejection sampling test has been successful. This results from the fact that the signature acts as a proof of knowledge only on the $\mathbf{s}$ part of the private key and not on the error $\mathbf{e}$. Nonetheless, thanks to the rounding, the

verifier will be to give a correct input to the hash function if the commitment is 'well-rounded'. Since not well-rounded signatures would leak information on the private key, this verification has to be performed in masked form.



Check the Rounding: This gadget denoted WRnd-Coeff in its coefficient version and WRnd in its polynomial version takes an arithmetic mod q input $(x_i)_{0\leq i\leq d}$ and 3 unmasked values $\texttt{param}_1$, $\texttt{param}_2$ and $\texttt{Tail}$ and outputs 1 iff $|\sum_i x_i \mod q| < \texttt{param}_1$ and $|\lfloor \sum_i x_i \mod q \rceil_{\texttt{Lst}}| < \texttt{param}_2$ where $\lfloor\cdot\rceil_{\texttt{Lst}}$ denotes keeping the $\texttt{Tail}$ least significant bits. It is presented in Gadget 21 and Gadget 22.

While the cost of this rather simple operation is negligible compared to polynomial multiplication in the unprotected signature, this test is fairly expensive in masked form. Indeed, it requires four comparisons in addition to the extraction of the low bits of x. The techniques are inspired from RS and Rnd gadgets. After trying the four comparisons method, we realized that the best strategy was actually to compute both absolute values with the sec|.| gadget. While comparisons only require one sec_+ and one shift, which is less than sec|.|, the cost of all sec& operations between the results of those comparisons makes our approach of computing the absolute value slightly better.

Gadget 21 — Masked well-rounded for integers (WRnd-Coeff)

Data: Integer $x \in \mathbb{Z}_q$ in arithmetic masked form $(x_i)_{0\leq i\leq d}$, $\texttt{param}_1$, $\texttt{param}_2$ and $\texttt{Tail}$ the number of least significant bits that are kept.
Result: $wr = 1$ iff $(|x| \leq \texttt{param}_1) \wedge (|\lfloor x \rceil_{\texttt{Lst}}| \leq \texttt{param}_2)$, $wr = 0$ otherwise.

1 $(\mathrm{SUP1}_i)_{0\leq i\leq d} \coloneqq (\texttt{param}_1, 0, ..., 0)$
2 $(\mathrm{SUP2}_i)_{0\leq i\leq d} \coloneqq (\texttt{param}_2, 0, ..., 0)$
3 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{A}_q{\rightsquigarrow}\mathsf{B}(x_i)_{0\leq i\leq d}$
4 $(a_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}|.|((x'_i)_{0\leq i\leq d}, \lceil \log_2(q) \rceil)$
5 $(a_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+((a_i)_{0\leq i\leq d}, (\mathrm{SUP1}_i)_{0\leq i\leq d}))$
6 $(b_i)_{0\leq i\leq d} \coloneqq (a_i)_{0\leq i\leq d} \gg (\mathrm{RADIX} - 1)$
7 $(x'_i)_{0\leq i\leq d} \leftarrow \mathsf{Ref}_{\oplus}((x'_i)_{0\leq i\leq d})$
8 $(x'_i)_{0\leq i\leq d} \coloneqq (x'_i)_{0\leq i\leq d} \wedge 2^{\texttt{Tail}} - 1$
9 $(y_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}|.|((x'_i)_{0\leq i\leq d}, \texttt{Tail})$
10 $(y_i)_{0\leq i\leq d} \leftarrow \mathsf{sec}_+((y_i)_{0\leq i\leq d}, (\mathrm{SUP2}_i)_{0\leq i\leq d}))$
11 $(b'_i)_{0\leq i\leq d} \coloneqq (y_i)_{0\leq i\leq d} \gg (\mathrm{RADIX} - 1)$
12 $(b_i)_{0\leq i\leq d} \leftarrow \mathsf{sec\&}((b_i)_{0\leq i\leq d}, (b'_i)_{0\leq i\leq d})$
13 **return** $wr \coloneqq \mathsf{Full}{\oplus}((b_i)_{0\leq i\leq d})$

Lemma 28. *The gadget* WRnd-Coeff *in Gadget 21 is d-MI secure.*

Proof: A graphical representation of Gadget 21 is in Fig. A.6. Let $\delta \leq d$ be the number of observations made by the attacker. Our goal is to prove that all these δ observations can be perfectly simulated with at most δ shares of $(x_i)_{0\leq i\leq d}$.
In the following, we consider the following distribution of the attacker's δ observations: δ_1 observed during the computation of $\mathsf{A}_q{\rightsquigarrow}\mathsf{B}$ that produces shares of $(x'_i)_{0\leq i\leq d}$, δ_2 observed during the computation of the upper sec|.| that produces the shares of $(a_i)_{0\leq i\leq d}$, δ_3 observed during the Refresh, δ_4 observed during the computations of the $\wedge$ and sec|.| that produces the shares of $(y_i)_{0\leq i\leq d}$, δ_5 observed during the sec_+ that produces $(a_i)_{0\leq i\leq d}$, δ_6 observed during the sec_+ that produces

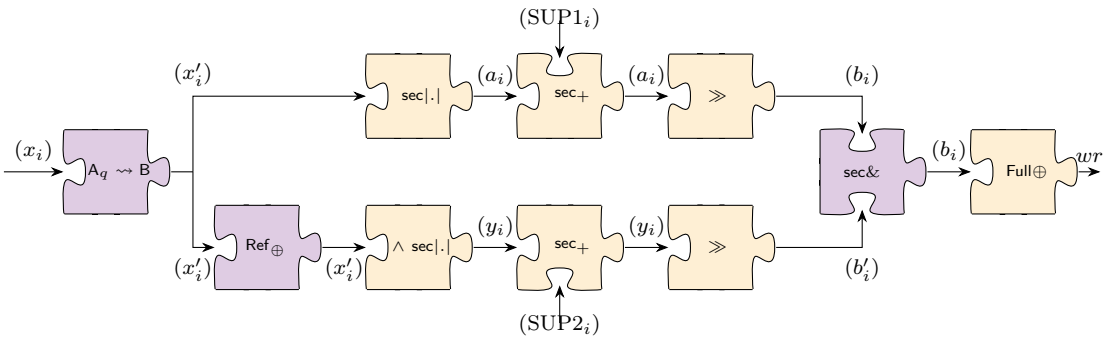


Figure A.6: *Masked* WRnd-Coeff *structure*

$(y_i)_{0\leq i\leq d}$, δ_7 observed during the shift step that produces $(b_i)_{0\leq i\leq d}$, δ_8 observed during the shift step that produces $(b'_i)_{0\leq i\leq d}$, δ_9 observed during the sec&, and finally δ_{10} observed during the final Full⊕ step. Finally, we have $\sum_{i=1}^{10}\delta_i \leq \delta$.

We build the proof from right to left. The algorithm Full⊕ is d-NI. As a consequence, all the observations from its call can be perfectly simulated with at most $\delta_{10} \leq \delta$ shares of b. The sec& algorithm is d-SNI, thus $d-$ NI secure. So, all the observations from its call can be perfectly simulated with at most $\delta_9 + \delta_{10} \leq \delta$ shares of b and b'. If we look at the lower gadgets of the figure, let us consider the shift that creates b', the sec_+ that creates y and the $\wedge, \mathsf{sec}|.|$. All three gadgets are d-NI secure, so all observations at the right side of $\wedge, \mathsf{sec}|.|$ can be simulated with at most $\delta_4 + \delta_6 + \delta_8 + \delta_9 + \delta_{10} \leq \delta$ share of x'. We now consider the Refresh algorithm. Since it is d-SNI secure and since the output and local observations are still less than δ, all observations from its call can be perfectly simulated with at most $\delta_3 \leq \delta$ shares of x'. Now let us consider the upper gadgets. The shift that creates b, the sec_+ that creates a and the $\mathsf{sec}|.|$ are d-NI secure, so all observations at the right side of $\mathsf{sec}|.|$ can be simulated with at most $\delta_2 + \delta_5 + \delta_7 + \delta_9 + \delta_{10} \leq \delta$ shares of x'. Finally, we consider the algorithm $\mathsf{A}_{q}\!\rightsquigarrow\!\mathsf{B}$ which is d-NI secure. There are at most $\delta_3 + (\delta_2 + \delta_5 + \delta_7 + \delta_9 + \delta_{10}) \leq \delta$ observations made on the outputs and δ_1 made locally. Thus, all the observations during WRnd-Coeff can be simulated with at most $\delta_1 + \delta_2 + \delta_3 + \delta_5 + \delta_7 + \delta_9 + \delta_{10} \leq \delta \leq d$ shares of the input x. □

Gadget 22 — Masked well-rounded (WRnd)

Data: A shared polynomial $(\mathbf{x}_i)_{0\leq i\leq d}$, in mod-$q$ arithmetic masked representation, $\mathtt{param}_1$, $\mathtt{param}_2$ and $\mathtt{Tail}$ the number of least significant bits that are kept.

Result: The bit wr equal to 1 if all the coefficients, denoted $x^{(i)}$ of $\mathbf{x}$ satisfy $(|x^{(i)}| \leq \mathtt{param}_1) \wedge (|\lfloor x^{(i)} \rceil_{\mathtt{Lst}}| \leq \mathtt{param}_2)$

for $j = 0$ *to* $n-1$ **do**
  $wr^{(j)} \leftarrow \mathsf{WRnd\text{-}Coeff}(x^{(j)}, \mathtt{param}_1, \mathtt{param}_2, \mathtt{Tail})$
end
return $wr := wr^{(0)} \wedge wr^{(1)} \cdots \wedge wr^{(n-1)}$

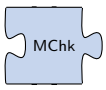
As with the previous large gadgets, the WRnd gadget can be generalized for polynomials in $\mathbb{Z}/q\mathbb{Z}$ by applying WRnd-Coeff to each of the coefficients. The values $wr^{(j)}$

correspond to the conditions of the well rounding, and more precisely, the positions of the coefficients of the polynomial that do not pass the condition. From Remark 5 and Lemma 28, we can make the following corollary.

Corollary 5. *Gadget* WRnd *is d-NIo secure with public outputs* $wr^{(j)}$.

▶ Masked Checking

The purpose of this gadget is to check that the sum of the h most significant coefficients of its inputs is not too large. To recover the most significant coefficients, this algorithm uses a straightforwardly masked bubble sort by doing h passes on the list of coefficients. The bubble sort uses a masked exchange subroutine where the bit 0 or 1, representing the need for an exchange or not, is also masked. It finishes with a masked comparison with a precomputed bound.



Masked checking: This gadget denoted MChk takes an arithmetic mod q input $(\mathbf{x}_i)_{0\le i\le d}$, an unmasked integer h and a an unmasked bound S. It outputs a bit ms such that $ms = 1$ iff the sum of the h largest coefficients of $\mathbf{x}$ is larger than S, $ms = 0$ otherwise. It is presented in Gadget 23.

Lemma 29. *Assuming that the bubble sort in step 2 is a d-NI subgadget, the gadget* MaskedCheck *in Gadget 23 is d-NI secure.*

Proof: Gadget 23 is a succession of non interfering gadgets with no cycles and the last gadget is d-NI. Thus, the whole algorithm is d-NI secure. □

Gadget 23 — Masked Check (MChk)

Data: An arithmetic masking of a polynomial $(\mathbf{x}_i)_{0\le i\le d}$, an integer h and a bound S

Result: $ms := 1$ if the sum of its h largest coefficients is larger than the bound S and 0 otherwise

1 $(\mathrm{BOUND}_i)_{0\le i\le d} := (-S, 0, \ldots, 0)$

2 Find the h largest coefficients $((c_i^{(0)})_{0\le i\le d}, \ldots, (c_i^{(h-1)})_{0\le i\le d})$ of $(\mathbf{x}_i)_{0\le i\le d}$ with a masked bubble sort.

3 $(sum_i)_{0\le i\le d} \leftarrow \mathsf{sec}_+((c_i^{(0)})_{0\le i\le d}, \ldots, (c_i^{(h-1)})_{0\le i\le d})$

4 $(\delta_i)_{0\le i\le d} \leftarrow \mathsf{sec}_+((sum_i)_{0\le i\le d}, (\mathrm{BOUND}_i)_{0\le i\le d})$

5 $(\delta_i)_{0\le i\le d} := ((\delta_i)_{0\le i\le d} \gg RADIX - 1)$

6 **return** $ms := \mathsf{Full}\oplus(\delta)$

RÉSUMÉ

La cryptographie fondée sur les réseaux euclidiens représente une alternative prometteuse à la cryptographie asymétrique utilisée actuellement, en raison de sa résistance présumée à un ordinateur quantique universel. Cette nouvelle famille de schémas asymétriques dispose de plusieurs atouts parmi lesquels de fortes garanties théoriques de sécurité, un large choix de primitives et, pour certains de ses représentants, des performances comparables aux standards actuels. Une campagne de standardisation post-quantique organisée par le NIST est en cours et plusieurs schémas utilisant des réseaux euclidiens font partie des favoris. La communauté scientifique a été encouragée à les analyser car ils pourraient à l'avenir être implantés dans tous nos systèmes. L'objectif de cette thèse est de contribuer à cet effort.

Nous étudions la sécurité de ces nouveaux cryptosystèmes non seulement au sens de leur résistance à la cryptanalyse en "boîte noire" à l'aide de moyens de calcul classiques, mais aussi selon un spectre plus large de modèles de sécurité, comme les attaques quantiques, les attaques supposant des failles d'utilisation, ou encore les attaques par canaux auxiliaires. Ces différents types d'attaques ont déjà été largement formalisés et étudiés par le passé pour des schémas asymétriques et symétriques pré-quantiques. Dans ce mémoire, nous analysons leur application aux nouvelles structures induites par les réseaux euclidiens. Notre travail est divisé en deux parties complémentaires : les contremesures et les attaques.

La première partie regroupe nos contributions à l'effort actuel de conception de nouvelles protections algorithmiques afin de répondre aux nombreuses publications récentes d'attaques par canaux auxiliaires. Les travaux réalisés en équipe auxquels nous avons pris part on abouti à l'introduction de nouveaux outils mathématiques pour construire des contre-mesures algorithmiques, appuyées sur des preuves formelles, qui permettent de prévenir systématiquement les attaques physiques et par analyse de temps d'exécution. Nous avons ainsi participé à la protection de plusieurs schémas de signature fondés sur les réseaux euclidiens comme GLP, BLISS, qTesla ou encore Falcon.

Dans une seconde partie consacrée à la cryptanalyse, nous étudions dans un premier temps de nouvelles attaques qui tirent parti du fait que certains schémas de chiffrement à clé publique ou d'établissement de clé peuvent échouer avec une faible probabilité. Ces échecs sont effectivement faiblement corrélés au secret. Notre travail a permis d'exhiber des attaques dites « par échec de déchiffrement » dans des modèles de failles d'utilisation ou des modèles quantiques. Nous avons d'autre part introduit un outil algorithmique de cryptanalyse permettant d'estimer la sécurité du problème mathématique sous-jacent lorsqu'une information partielle sur le secret est donnée. Cet outil s'est avéré utile pour automatiser et améliorer plusieurs attaques connues comme des attaques par échec de déchiffrement, des attaques classiques ou encore des attaques par canaux auxiliaires.

MOTS CLÉS

Cryptographie ★ Cryptanalyse ★ Réseaux euclidiens

ABSTRACT

Lattice-based cryptography is considered as a quantum-safe alternative for the replacement of currently deployed schemes based on RSA and discrete logarithm on prime fields or elliptic curves. It offers strong theoretical security guarantees, a large array of achievable primitives, and a competitive level of efficiency. Nowadays, in the context of the NIST post-quantum standardization process, future standards may ultimately be chosen and several new lattice-based schemes are high-profile candidates. The cryptographic research has been encouraged to analyze lattice-based cryptosystems, with a particular focus on practical aspects. This thesis is rooted in this effort.

In addition to black-box cryptanalysis with classical computing resources, we investigate the extended security of these new lattice-based cryptosystems, employing a broad spectrum of attack models e.g. quantum, misuse, timing or physical attacks. Accounting that these models have already been applied to a large variety of pre-quantum asymmetric and symmetric schemes before, we concentrate our efforts on leveraging and addressing the new features introduced by lattice structures. Our contribution is twofold: defensive, i.e. countermeasures for implementations of lattice-based schemes and offensive, i.e. cryptanalysis.

On the defensive side, in view of the numerous recent timing and physical attacks, we wear our designer's hat and investigate algorithmic protections. We introduce some new algorithmic and mathematical tools to construct provable algorithmic countermeasures in order to systematically prevent all timing and physical attacks. We thus participate in the actual provable protection of the GLP, BLISS, qTesla and Falcon lattice-based signatures schemes.

On the offensive side, we estimate the applicability and complexity of novel attacks leveraging the lack of perfect correctness introduced in certain lattice-based encryption schemes to improve their performance. We show that such a compromise may enable decryption failures attacks in a misuse or quantum model. We finally introduce an algorithmic cryptanalysis tool that assesses the security of the mathematical problem underlying lattice-based schemes when partial knowledge of the secret is available. The usefulness of this new framework is demonstrated with the improvement and automation of several known classical, decryption-failure, and side-channel attacks.

KEYWORDS

Cryptography ★ Cryptanalysis ★ Lattices